Jül-4366

Peter Grünberg Institute / Institute of Complex Systems
Bioelectronics (PGI-8/ICS-8)

# Display and data analysis of signals from bioelectronic devices

*Khaled Abdel-Latif*

Member of the Helmholtz Association

JÜLICH
FORSCHUNGSZENTRUM

# Display and data analysis of signals from bioelectronic devices

*Khaled Abdel-Latif*

Berichte des Forschungszentrums Jülich; 4366
ISSN 0944-2952
Peter Grünberg Institute / Institute of Complex Systems
Bioelectronics (PGI-8/ICS-8)
Jül-4366

Vollständig frei verfügbar im Internet auf dem Jülicher Open Access Server (JUWEL)
unter http://www.fz-juelich.de/zb/juwel

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek, Verlag
D-52425 Jülich · Bundesrepublik Deutschland
☎ 02461 61-5220 · Telefax: 02461 61-6103 · e-mail: zb-publikation@fz-juelich.de

# Abstract

The aim of this thesis was to develop the means to display and analyze cellular signals acquired by chip-based extracellular recordings. Two types of chips were used for these measurements: the Multi-Electrode Array (MEA) and the Silicon-based Field-Effect Transistor (FET). The extracellular measurements are based upon an in-house built and programmed low-noise amplifier system. Using the chips and hardware to amplify the signals, the display process and analysis of the signals could be achieved. "Viewer" was a program performed in "LabVIEW" that enabled the measured channels from a MEA chip to be displayed. In this program, the signals are imported and then each channel is displayed individually on the graphs. Exporting the data was an important part of the program. Finally, the efficiency of the program was tested.

# Kurzfassung

Das Ziel dieser Arbeit war die Schaffung der Voraussetzungen zur Darstellung und Analyse der Daten von sensor-chips durch extrazelluläre Aufnahmen. Es gibt zwei Arten von Chips für die Messung, die MEA (Multi-Elektroden-Array) und den FET (Silicon-based Field-Effect Transistor). Die extrazelluläre Aufnahmen bassieren auf einem "low-noise amplifier"-System. Mit Hilfe der Chips und der Verwendung von Hardware, um die Signale zu verstärken, kann der Prozess der Signalaufnahme verfolgt werden. Der "Viewer" ist ein LabVIEW-Programm und dient der Darstellung von gemessenen Kanälen eines Multi-Elektrode-Array-Chips (MEA). In diesem Programm werden die Signale eingelesen und dann wird jeder Kanal in den Diagrammen dargestellt. Die Export-Funktion der Daten ist ein wichtiger Teil des Programms. Schließlich wurde die Effizienz des Programms getestet.

# Contents

**1. Introduction ..... 1**
**2. Theoretical Background ..... 3**
**2.1 Biological Background ..... 3**
2.1.1 Cell Membrane ..... 3
2.1.2 Action Potential ..... 4
**2.2 Background of Electrical Devices ..... 8**
2.2.1 Multi-Electrode Array ..... 8
2.2.2 Si-based Field-Effect Transistor ..... 9
**2.3 Software Background ..... 9**
2.3.1 LabVIEW ..... 9
2.3.2 Python ..... 11
**3. Materials and Methods ..... 14**
**3.1 Devices ..... 14**
**3.2 Materials ..... 17**
**4. Results and Discussion ..... 18**
**4.1 The Viewer ..... 18**
4.1.1 Displaying Missing Channels ..... 19
4.1.2 Efficient Data Handling ..... 24
4.1.3 Data Export ..... 30
4.1.4 Selecting Activated Channels ..... 36
**4.2 Extracellular Recording from Cardiac Cells ..... 42**
**4.3 PyAnalyzer ..... 48**
**5. Conclusions and Outlook ..... 52**
**5.1 Conclusion ..... 52**
**5.2 Outlook ..... 53**
**Bibliography ..... 55**
**Appendix A: The Viewer ..... 58**

**Appendix B: Python Code.....71**
**Appendix C: Cell Culture for HL-1 cell-line .....73**
**Appendix D: The Edwin Smith Surgical Papyrus.....74**
**Acknowledgements.....76**

# Abbreviations

| | |
|---|---|
| Ag | Silver |
| AgCl | Silver Chloride |
| AP | Action Potential |
| ASKII | American Standard Code Information Interchange |
| Ca | Calcium |
| DLL | Dynamic-Link-Library |
| ECM | Extracellular Matrix |
| EtOH | Ethyl Alcohol |
| FET | Field-Effect Transistor |
| FFT | Fast-Fourier Transform |
| FWHM | Full-Width-Half-Maximum |
| GUI | Graphical User Interface |
| I | Iterations (LabVIEW) |
| IC | Integrated Circuit |
| IDE | Integrated development environment |
| ISFET | Ion-Sensitive Field-Effect Transistor |
| K | Potassium |
| KB | Kilobyte |
| MB | Megabyte |
| GB | Gigabyte |
| MEA | Multi-Electrode Array |
| ml | Milliliter |
| µl | Microliter |

| | |
|---|---|
| mV | Millivolt |
| µV | Microvolt |
| Na | Sodium |
| NR | Nano-Ribbons |
| PCB | Printed Circuit Board |
| PDMS | Polydimethylsiloxane |
| RMS | Root-Mean-Square |
| SubVI | Sub-Virtual Instrument (LabVIEW) |

# Table of Figures

Figure 1: Cross section of an animal cell [5]. 3
Figure 2: Freeze-fracture technique splits plasma membrane [6]. 4
Figure 3: Occurrence of an action potential in a neuron (resting potential, depolarization, repolarization and refractory period) [5]. 5
Figure 4: The process of the sodium potassium pump [6]. 6
Figure 5: Example of the structure of a neuron [5]. 7
Figure 6: Unidirectional signal transmission from a presynaptic to a postsynaptic cell by a chemical synapse [7]. 7
Figure 7: Photo of a MEA chip. 8
Figure 8: Schematic of the flip-flop procedure [8]. 8
Figure 9: The Ion-Sensitive Field-Effect Transistor (ISFET) [10]. 9
Figure 10: An example for a front panel. 10
Figure 11: An example for a block diagram. 11
Figure 12: An example of a "SubVI" in a main program. 11
Figure 13: An example of "Spyder IDE". 12
Figure 14: An example of "Python QT (Quick Time)". 12
Figure 15: Non-interpolated heart rate and spectrogram obtained with "Matplotlib" [14]. 13
Figure 16: Photograph of the stereo-optical microscope. 14
Figure 17: Schematic of extra-cellular recordings and data analysis. 15
Figure 18: Schematic of the electronic readout and control of the amplifier setup. 16
Figure 19: Photograph of a front panel (a) [8]. 16
Figure 20: Photograph of a back panel (b) [8]. 16
Figure 21: An example of a cardio myocyte-like cell [19]. 17
Figure 22: Overview of the viewer program. 19
Figure 23: Last channel 64 in the lower right corner was missing. 20
Figure 24: "Read from file SubVI". 20
Figure 25: Comparison between active channels and iteration number. 21
Figure 26: Solving the problem of the wrong arrangement of the channels. 22
Figure 27: Last channel 64 was displayed correctly. 22
Figure 28: Property node for max/min functions. 22
Figure 29: "Value change of the graphs" in the disable state (the "True" case). 23
Figure 30: The "False" case when the indicator is turned off. 23
Figure 31: An example of disabling the channel 29 in the front panel. 24
Figure 32: Enabling the graphs in the "True" case. 25
Figure 33: Disabling the graphs in the "False" case. 25
Figure 34: Enabling the functions in the "all in one graph" chart. 25
Figure 35: Disabling the functions in the "all in one graph" chart. 26

Figure 36: Charts disabled while running the program. 26
Figure 37: Charts enabled while running the program. 27
Figure 38: "Add disabled channels SubVI". 27
Figure 39: "Read from file to graph SubVI". 28
Figure 40: Non-measured channel indicator (in green). 28
Figure 41: Taking the non-measured channel indicator as an output of the "add non-measured channels SubVI". 29
Figure 42: Connecting the signal from "read binary file SubVI" to the "read from file to graph SubVI". 29
Figure 43: The new structure of the "read from file to graph SubVI". 30
Figure 44: Export function in the older initial version. 31
Figure 45: Updated export functions without headers. 31
Figure 46: The structures of the export function with headers. 32
Figure 47: Exported file (channel 8 disabled) in the older version. 32
Figure 48: "Export file without headers" after using the updated version of the export function. 33
Figure 49: "Export file with headers" after using the updated version of the export function. 33
Figure 50: General overview of the headers function. 34
Figure 51: Memory consumption of different files sizes in a 32-bit LabVIEW environment. 35
Figure 52: Memory consumption for different files sizes in 64-bit LabVIEW environment. 36
Figure 53: Control list for the activated channels. 37
Figure 54: Control list and the corresponding buttons while running an input file with disabling the channels 1-8-57-64. 38
Figure 55: Control list while running the program by disabling channel 8. 39
Figure 56: Adding channels 4 and 5 to the selected channel 3. 40
Figure 57: Adding a gap between the channels 3, 4 and 5. 40
Figure 58: Selecting the channel 3 before auto-scaling. 41
Figure 59: Selecting channel 3 after auto-scaling. 41
Figure 60: Signals of Channel 37 displaying spontaneous activity. 42
Figure 61: Zooming in an action potential signal of channel 37 during spontaneous activity. 43
Figure 62: Channel 37 after pipetting 120 µl noradrenalin. 44
Figure 63: Channel 13 after pipetting 120 µl noradrenalin. 45
Figure 64: Zooming on channel 37 after pipetting 120 µl noradrenalin. 46
Figure 65: Zooming on channel 13 after pipetting 120 µl noradrenalin. 46
Figure 66: Schematic view of a MEA chip. 47
Figure 67: Applying the function "scipy.fft()" to the random noise signal. 49

Figure 68: Plot of the amplitude, sigma range and RMS value of the noise signal. .. 50
Figure 69: Plot of the Power-Spectral-Density (PSD) of a random noise function. ... 50
Figure 70: Python QT framework for the graphical interface of the PyAnalyzer. ..... 51
Figure 71: The notification (in red) for the user in the main program. ..... 53

# 1. Introduction

The ability of all living organisms to survive depends on the internal balance and the external environment. Their bodies apply a number of mechanisms to communicate the signals necessary for activity. In higher animals there are many systems responsible for this, one of them is the nervous system. It is responsible for sending signals from one cell to another or through different parts of the body. There are many ways that a cell can send signals to other cells. One is by releasing chemicals called hormones into the internal circulation, so that they can diffuse to distal sites. The other is producing an electrical current that travels rapidly down the axon of the nerve cell into the synaptic clefts [1].

The study of the nervous system, neuroscience, is important to understand the biological basis of thought, emotion, and behavior. The ancient Egyptians were responsible for the oldest written record using the word "brain" and provided the first written information about the anatomy of the brain. The word "brain" appears on an ancient paper-like document (a "papyrus") called the Edwin Smith Surgical Papyrus [2]. This document (see Appendix D) was written around the year 1700 BC and is considered the first medical document in the history of mankind.

The cell is the basic unit of living tissue. Cells are specialized in order to perform different functions. Most animal cells have a difference in electrical charge across the plasma membrane. Both nerve cells and muscle cells can produce electrochemical impulses and conduct them along the membrane. In muscle cells, this electric phenomenon is also associated with the contraction of the cell [3]. In both cell types, the membrane generates impulses as a result of excitation. These impulses move in both cell types in the same manner.

Cultured cardiac myocytes offer many advantages for developmental, physiological, and pharmacological studies of cardiac tissue because they allow for direct cell manipulation and control of environmental parameters without interference from the compensatory feedback mechanisms that exist in vivo [4].

This thesis deals with the development of the means to display and analyze the action potential signals of cardiac cells. These cells have been adjusted to be cultivated in a suitable culture medium and environment on a chip.

After amplifying the signals, the display process and analysis could be performed.

This is achieved by a program called “Viewer”. The program is run in LabVIEW and can display the signals of the 64 channels in case of the Multi-Electrode Array (MEA) chip. The aim of the project was to develop the program so that it is capable of displaying all the measured channels correctly and in the right orientation.

After displaying the data, an export function was installed to export the data for further analysis. This function enables the user to export the files with the corresponding headers so that it will be clear which channels are active.

In order to overcome the memory problems that occurred in LabVIEW, another program was introduced, which was written in “Python” programming language.

# 2. Theoretical Background

## 2.1 Biological Background

### 2.1.1 Cell Membrane

All living organisms have a basic functional and structural unit, which is the cell (Figure 1). It is the smallest living unit in our world. The origin of the word "cell" comes from the latin cella, meaning "small room."

The eukaryotic cells consist of a cell nucleus that encloses the chromosomes. The mitochondria are responsible for generating energy; they are considered as the power generators. The endoplasmic reticulum (ER) is the transport network and has two forms: the rough ER with ribosomes, and the smooth ER, which lacks ribosomes. In addition to that, there is also the "Golgi apparatus" which modifies macromolecules, for example proteins. Lysosomes digest the excess or damaged organelles. The centrosomes are very important for the cytoskeleton as they produce the microtubules, which are the key component of the cytoskeleton.

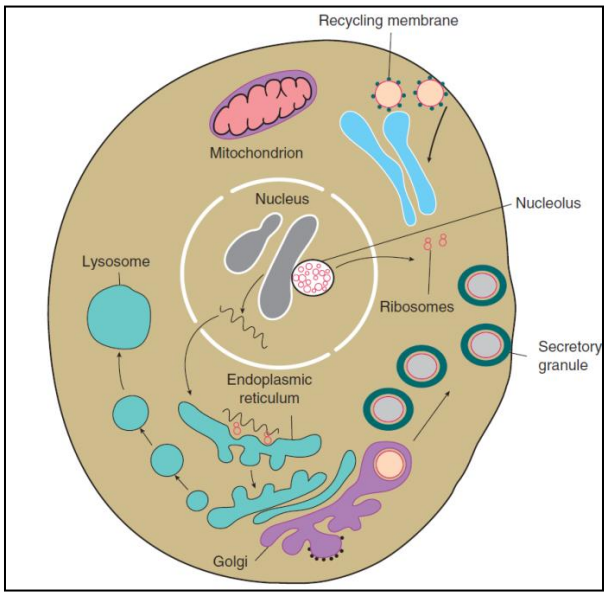



**Figure 1:** Cross section of an animal cell [5].

The cell is separated from the extracellular environment by a cell membrane, which is basically formed of a bilayer of phospholipids–amphiphilic molecules. These molecules contain a hydrophilic head and a hydrophobic tail. The unipolar tails build the center of the membrane. The hydrophilic heads are

pointing outwards. Ions and water cannot penetrate the membrane. This is due to the hydrophobicity of the inner layer [6]. A schematic diagram (Figure 2) is presented to show the freeze-fracture technique which is used to view a plasma membrane under a microscope.

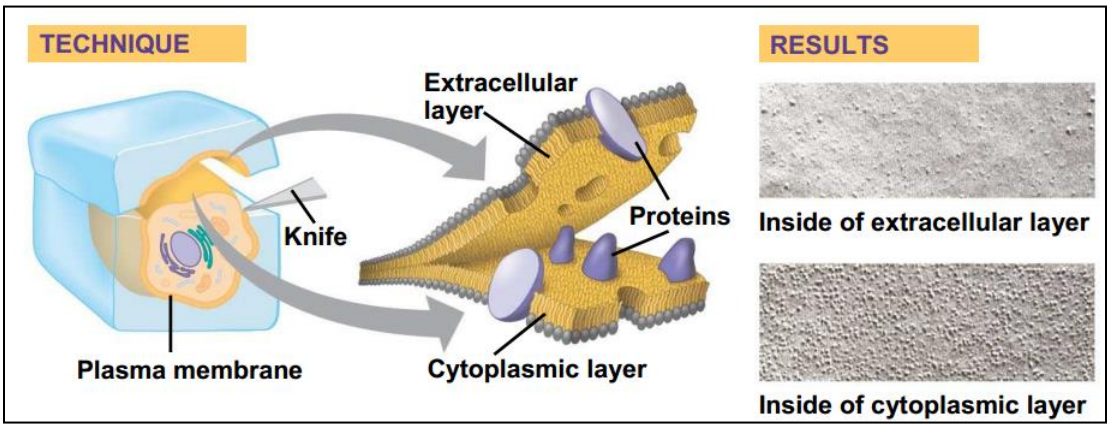



**Figure 2:** Freeze-fracture technique splits plasma membrane [6].

### 2.1.2 Action Potential

Action potentials (Figure 3) play a vital role in distributing and relaying information. These are ion-mediated electric pulses that can propagate along the membrane.

A neuronal cell has a resting potential in the range from -60 mV to -70 mV [7]. No signals can be transported during this resting potential. An action potential can occur only when a specific threshold voltage has been reached.

If the membrane potential starts to increase towards the positive values and reaches the threshold voltage (approximately -60 mV), the permeability of the cell membrane for sodium ions increases due to the opening of the voltage-gated sodium ion channels.

This process is called **depolarization,** in which sodium penetrates along the electrical gradient to the cell membrane. In the same period, the opening of the voltage-gated potassium channels occurs and this leads to the efflux of the potassium ions to the outside of the cell membrane.

After the closing of the sodium ion channels, there is no influx of the sodium ions into the cell. A new phase will occur which is the **repolarization** or the **hyperpolarization** due to the efflux of the potassium from the cell.

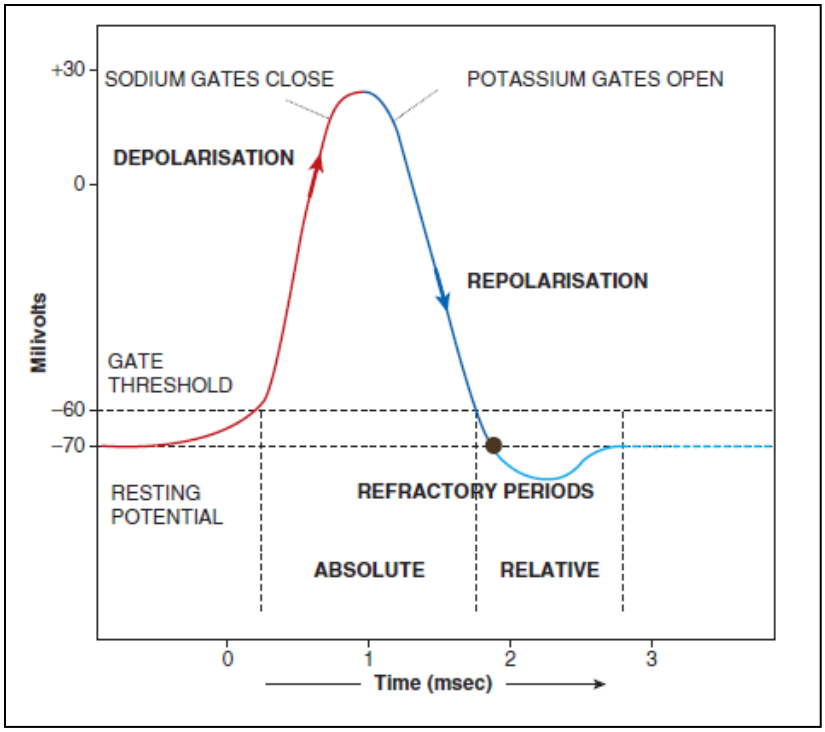



**Figure 3:** Occurrence of an action potential in a neuron (resting potential, depolarization, repolarization and refractory period) [5].

Consequently, the potassium channels slowly close, hence the sodium channels cannot open as well. This is called the **refractory period**. The initial resting potential will be reinstated. This is due to the diffusion by the sodium and potassium pump (Figure 4).

The mechanism of the sodium-potassium pump begins with the binding of the cytoplasmic $Na^+$ to the pump. This binding stimulates phosphorylation by ATP [6]. The sodium ions are expelled to the outside due to the change in the protein conformation. Then, the $K^+$ binds the protein and this leads to the release of the phosphate group. As soon as the phosphate group is released, the protein returns to the original conformation. As a result, $K^+$ is released, the $Na^+$ binds again, and the cycle is repeated.

The action potential follows **the all-or-nothing principle**, which indicates that a complete action potential can take place only when depolarization above the threshold voltage occurs.

The propagation of an action potential is of great importance. It is responsible for transportation and signal processing. The electronic conduction is responsible for the propagation of the action potential.

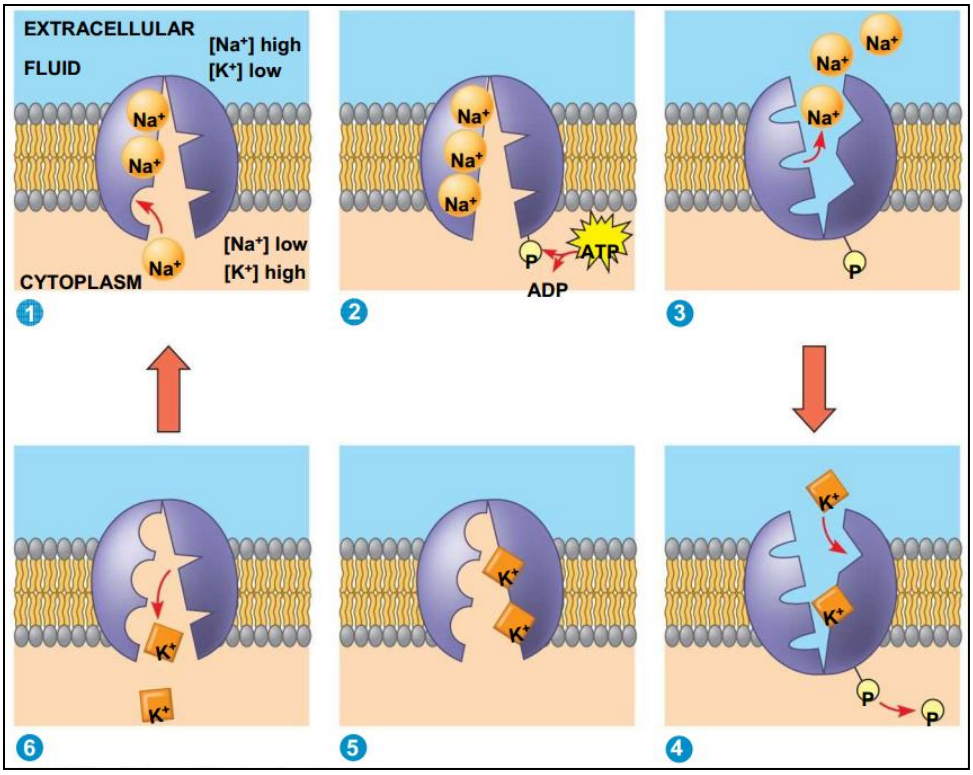



**Figure 4:** The process of the sodium potassium pump [6].

When a depolarization happens at one point of the axon in a neuron (Figure 5), this depolarization spreads passively to the areas that lie near the axon.

Due to this process, the nearby areas will reach the threshold, which will generate an action potential. This leads to an automatic spreading of the action potentials along the whole axon. As this process continues, the signal can be transported through the cell.

For the propagation of the signals between two neurons, the electrical signal is converted into chemical transmitters due to the chemical synapses (Figure 6). Two neurons are separated by a synaptic cleft (by chemical synapse) with the size of 20–40 nm [7]. As soon as an action potential arrives in the axon terminal of the presynaptic cell, the calcium channels open, which leads to an influx of $Ca^{2+}$ ions into the cell.

Increasing calcium concentration will lead to the fusion of the vesicles containing neurotransmitters with the presynaptic membrane. As a result, the neurotransmitters are released to the synaptic cleft and will diffuse to the other side to the membrane of the postsynaptic cell.

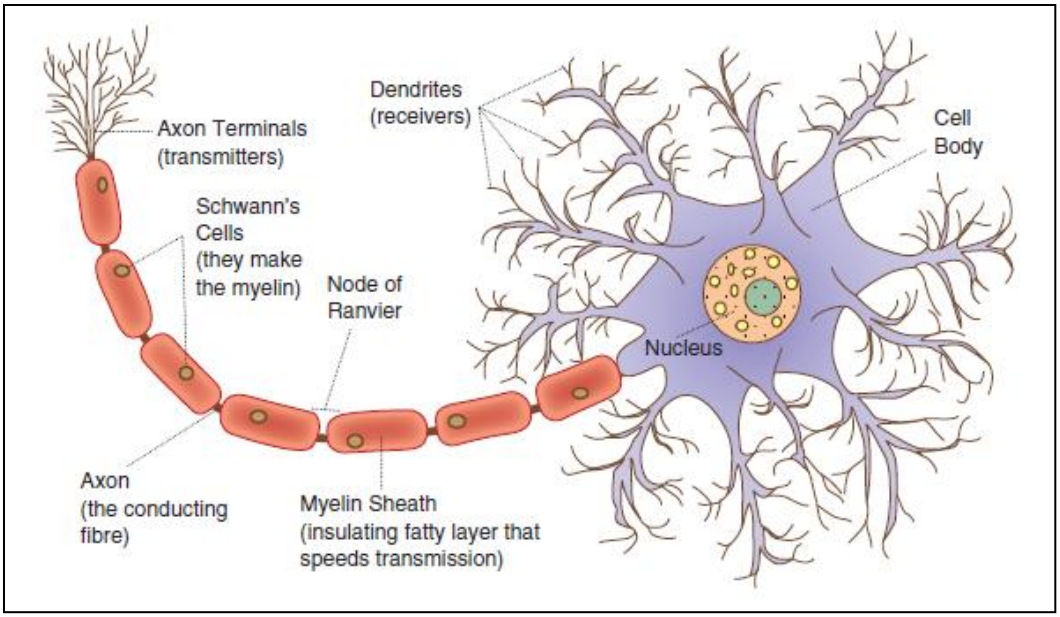



**Figure 5:** Example of the structure of a neuron [5].

Depending upon the kind of synapse, the channel either opens or closes and increases or decreases the conductance of the postsynaptic membrane. This causes the opening of ion channels. The influx of the positively charged ions leads to depolarization of the postsynaptic cell membrane, which in turn causes in this case the signal to be propagated.

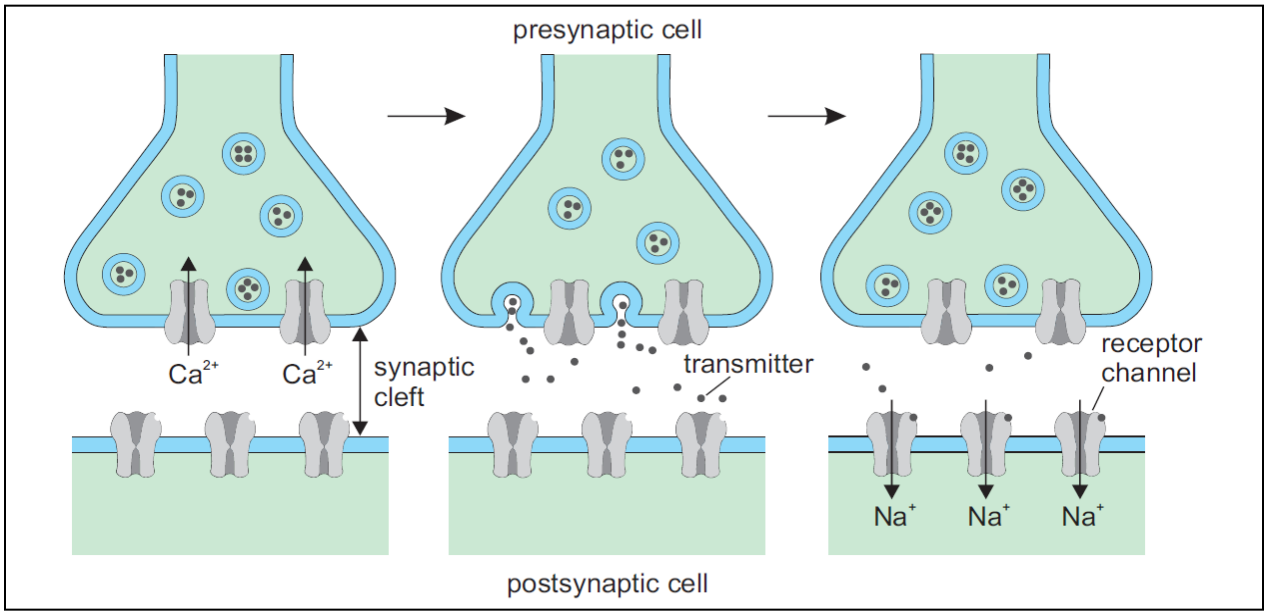



**Figure 6:** Unidirectional signal transmission from a presynaptic to a postsynaptic cell by a chemical synapse [7].

## 2.2 Background of Electrical Devices

### 2.2.1 Multi-Electrode Array

Multi-Electrode Arrays (MEAs) serve as one possible device to connect cells to an electronic circuitry. A MEA consists of passivated feed lines, which have selectively opened apertures to connect cells to the electronic circuitry where the signals (for example, action potentials) can be delivered. An example of a MEA chip (borofloat) is shown in (Figure 7).

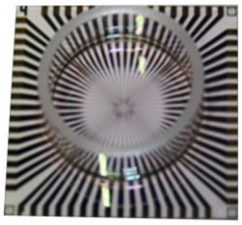


**Figure 7:** Photo of a MEA chip.

The construction of a MEA chip has many steps. One of the steps, which was contributed during the thesis in the labs is the flip-flop procedure (Figure 8). In this procedure, the goal is to establish a contact between the backside of the carrier (a) and the front side of the chip (b).

The bond pads of the carrier as well as the top side to the backside are glued with silver glue (c). After this step, the carrier is fastened with the chip [8]. The front view and the backside of the chip are shown in part (d) and (e).

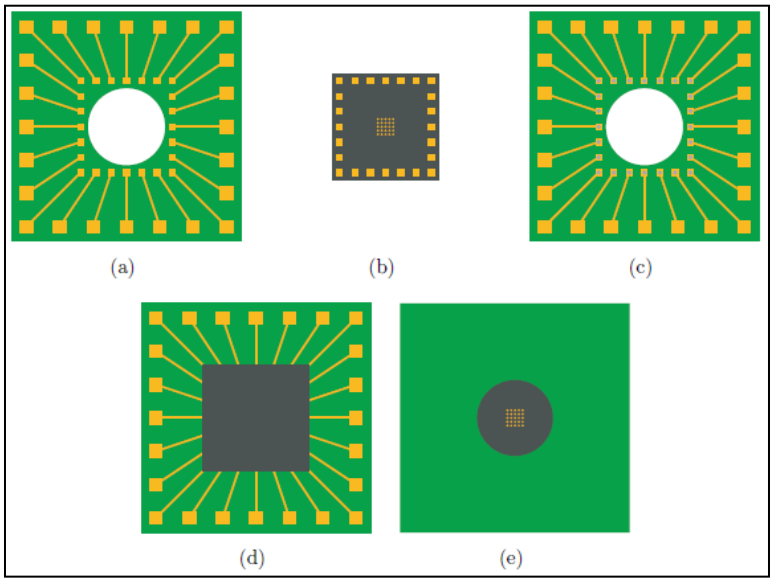



**Figure 8:** Schematic of the flip-flop procedure [8].

### 2.2.2 Si-based Field-Effect Transistor

The ISFET (Ion-Sensitive Field-Effect Transistor) was introduced in 1970 [9]. The typical structure of an ISFET is shown in Figure 9. There are two highly n-doped regions, the source contact and the drain contact. A p-doped region lies between the source and drain. There is also a gate-insulating layer and a sensor membrane. An electrolyte is in direct contact with the gate oxide, forming the gate. The gate voltage is applied across a reference electrode and immersed in the electrolyte.

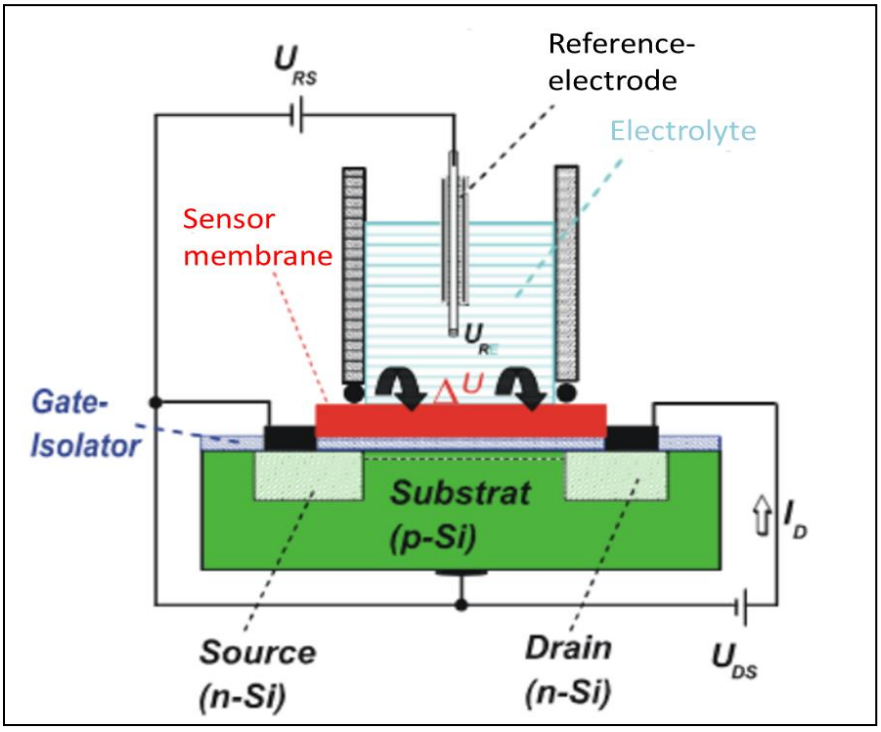



**Figure 9:** The Ion-Sensitive Field-Effect Transistor (ISFET) [10].

## 2.3 Software Background

### 2.3.1 LabVIEW

LabVIEW ("Laboratory Virtual Instrumentation Engineering Workbench") is a graphical programming language established by National Instruments (NI). This thesis used the English version "LabVIEW 2012" SP1, 32 bit & 64 bit.

LabVIEW consists of two windows, the "front panel", and the "block diagram" (Figure 10 and Figure 11). The front panel is the graphical interface for the user. The block diagram is for programming. In the front-panel example there is the numeric input control. This enables the user to input the desired

number and increment or decrement to other numbers. The Booelan control has two conditions, the "True" state where it is on and the "False" state where it is off. The gauge is an example of an indicator that shows the result in the graph.

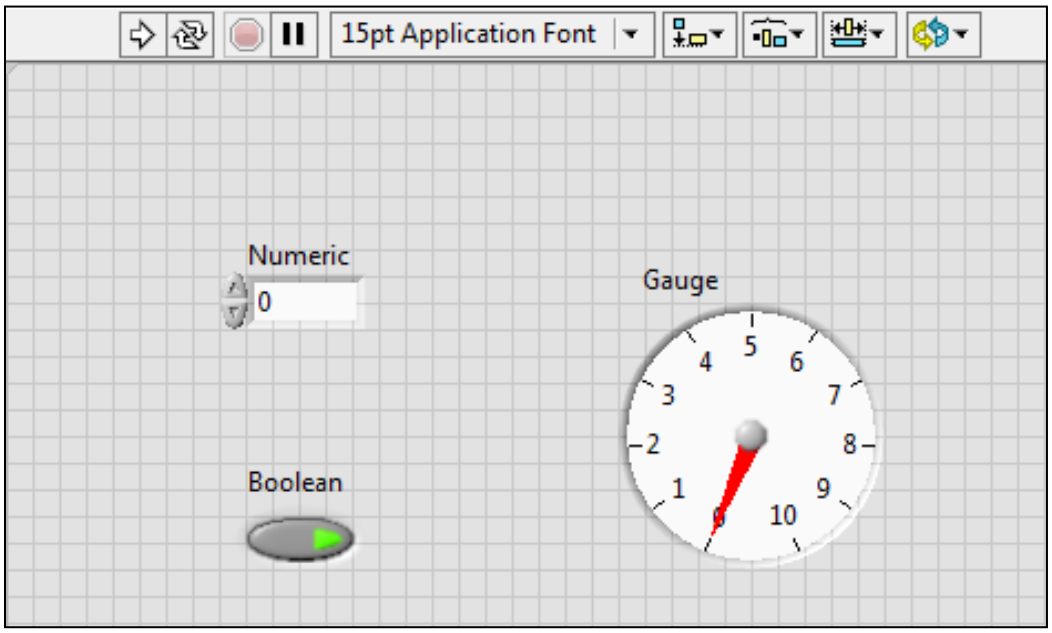



**Figure 10:** An example for a front panel.

The wiring between the objects takes place in the block diagram. The numeric input is being added to a constant number (Figure 11) and then connected to the gauge indicator. Therefore, the gauge will show the result of the addition. The frame surrounding the three objects is called a case structure, which has two conditions: the "True" state and the "False" state. The choice between these two states is controlled by the input of the case structure in the small green box with the "?" sign. If the "Boolean" input is "True", then the "True" case will be activated and vice versa concerning the "False" case.

It is also very common to use the so-called virtual instruments (VI). Another useful structure is the "SubVI". The "SubVI" is an individual function and can be called by another VI. The "SubVI" is shown as a small box marked in orange in a main program (Figure 12). It has the capability to save more space so that the structure is less complicated. This achieves a compacter and clearer main program. The program LabVIEW was used to build the "Viewer" program, which analyses the data from the 64 channels in the MEA chips.

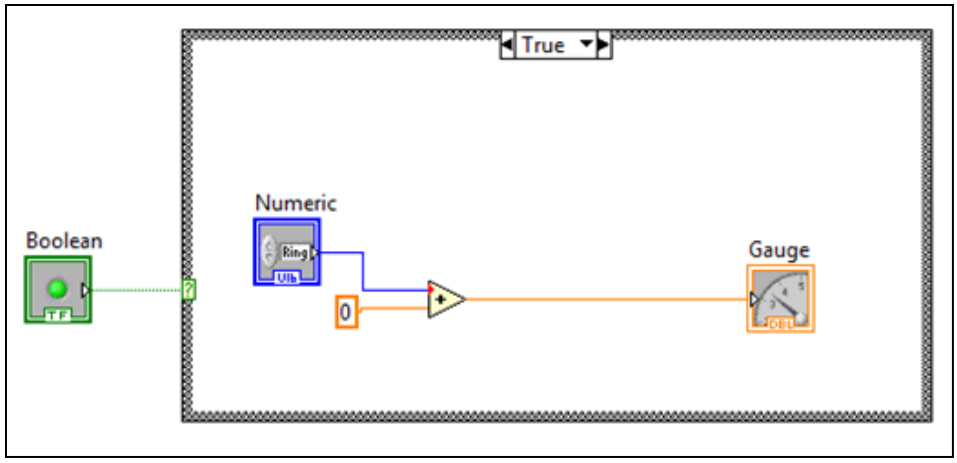



**Figure 11:** An example for a block diagram.

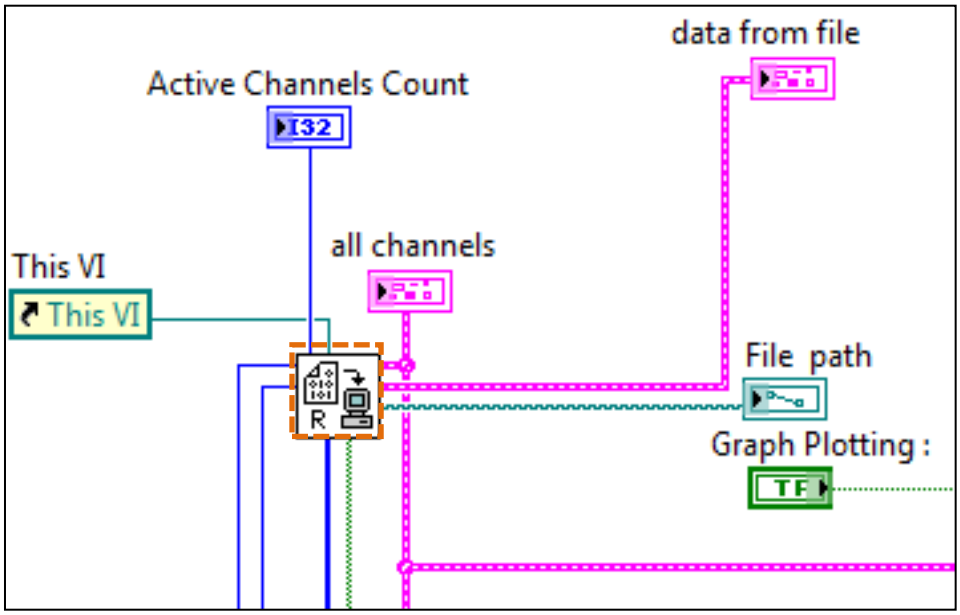



**Figure 12:** An example of a "SubVI" in a main program.

### 2.3.2 Python

Many problems occurred during the development of the "viewer". The biggest problem was the enormous usage of memory due to the graphs, for example, the waveform graph. This limited the user to importing input files with specific file range up to a certain maximum as will be discussed in more detail in **chapter 4.1.3**. This is one of the main reasons that led to the development of a program in "Python" programming language.

Python has many features, including scientific-related modules, like "Numpy" and "Scipy" as well as very useful integrated-development environments (IDE), such as "Spyder" [11] (Figure 13).

```
1 # -*- coding: utf-8 -*-
2 from __future__ import division
3 import scipy
4 import scipy.fftpack
5 import pylab
6 #from scipy import pi
7 """
8 Created on Fri Aug 09 09:58:44 2013
9
10 @author: k.abdel-latif
11 """
12 #from mpl_toolkits.mplot3d import Axes3D
13 from scipy import fft
14 import numpy as np
15 #import matplotlib.pyplot as plt
16 from math import sqrt
17 from numpy import mean
18 #import scipy
19 #import matplotlib.mlab as mlab
20
```

**Figure 13:** An example of "Spyder IDE".

Additionally, modules such as "Python QT" (Figure 14) enable the user to create a graphical interface [12]. The user can then plot graphs easily using "Matplotlib" [13] (Figure 15).

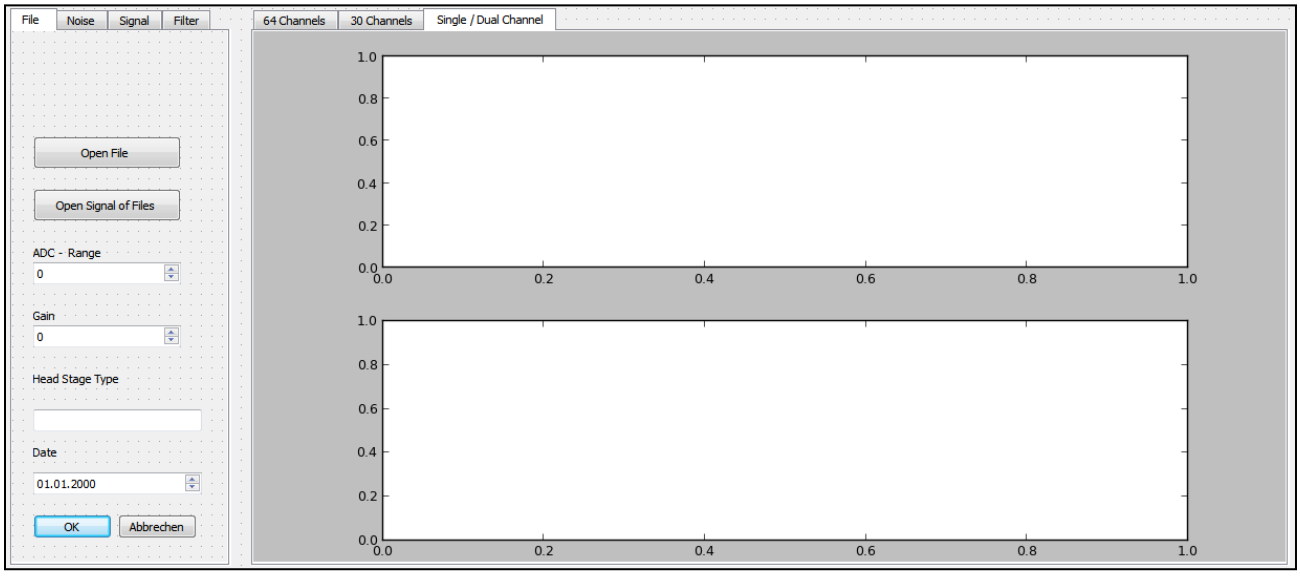



**Figure 14:** An example of "Python QT (Quick Time)".

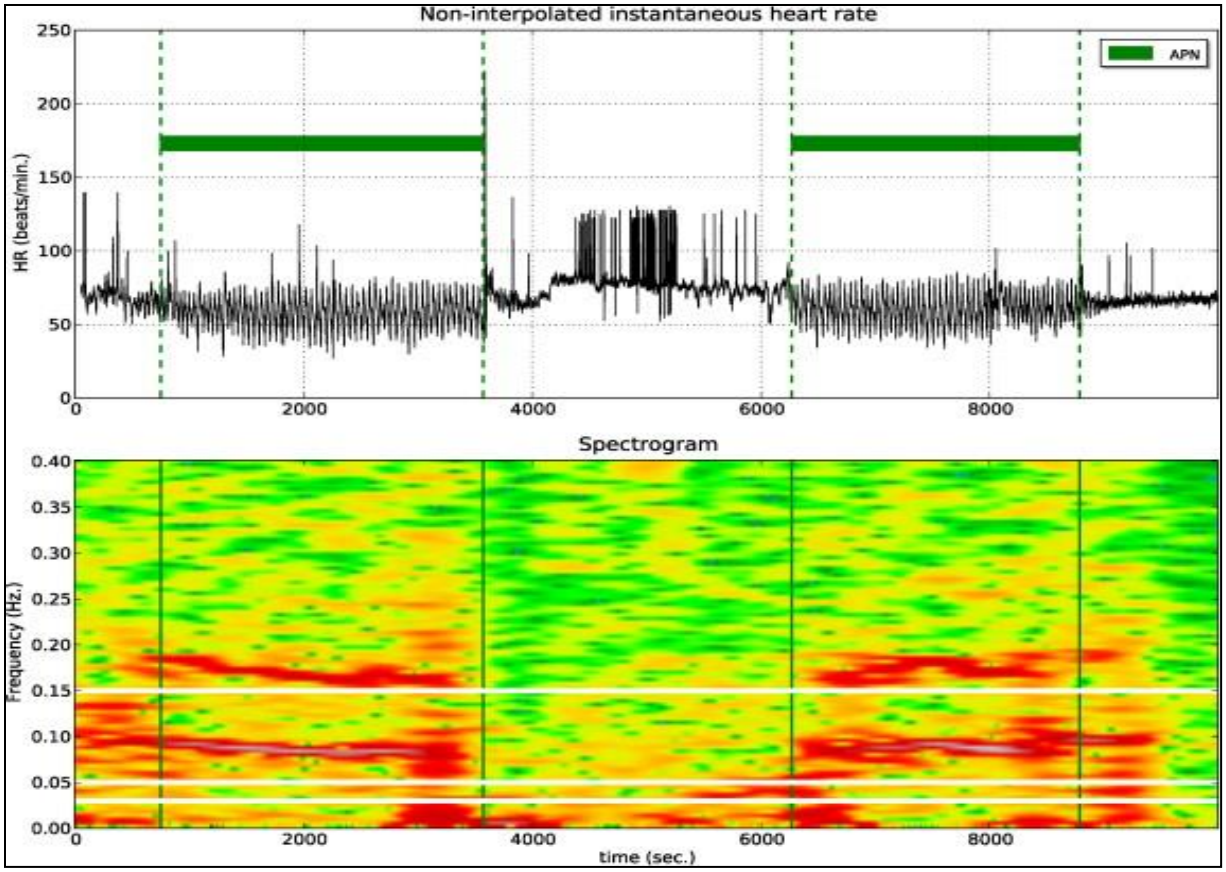



**Figure 15:** Non-interpolated heart rate and spectrogram obtained with "Matplotlib" [14].

# 3. Materials and Methods

## 3.1 Devices

To check the functionality of the chips, a stereo-optical microscope (Carl Zeiss Microimaging GmbH, Göttingen, Germany) was used (Figure 16). The process was monitored by a computer.

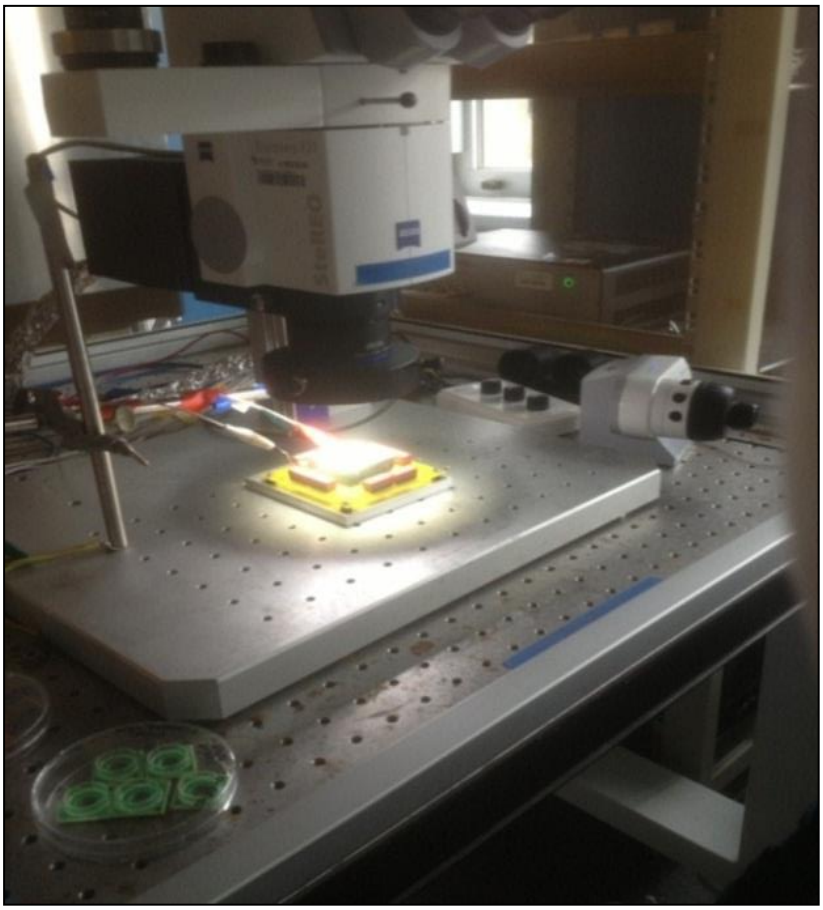


**Figure 16:** Photograph of the stereo-optical microscope.

It is also important to mention how the general process (Figure 17) of measuring the action potential (AP) occurred. For the experiment, an HL-1 cell line was used. They were derived from AT-1 cells [15] (mouse cardiac myocyte tumor).

The MEA chip was inserted into the socket of the head stage and an Ag/Agcl electrode was inserted to the medium. This Ag/Agcl electrode is the reference electrode, which gives a reference potential and grounding. This is a single-ended measurement, not a differential measurement.

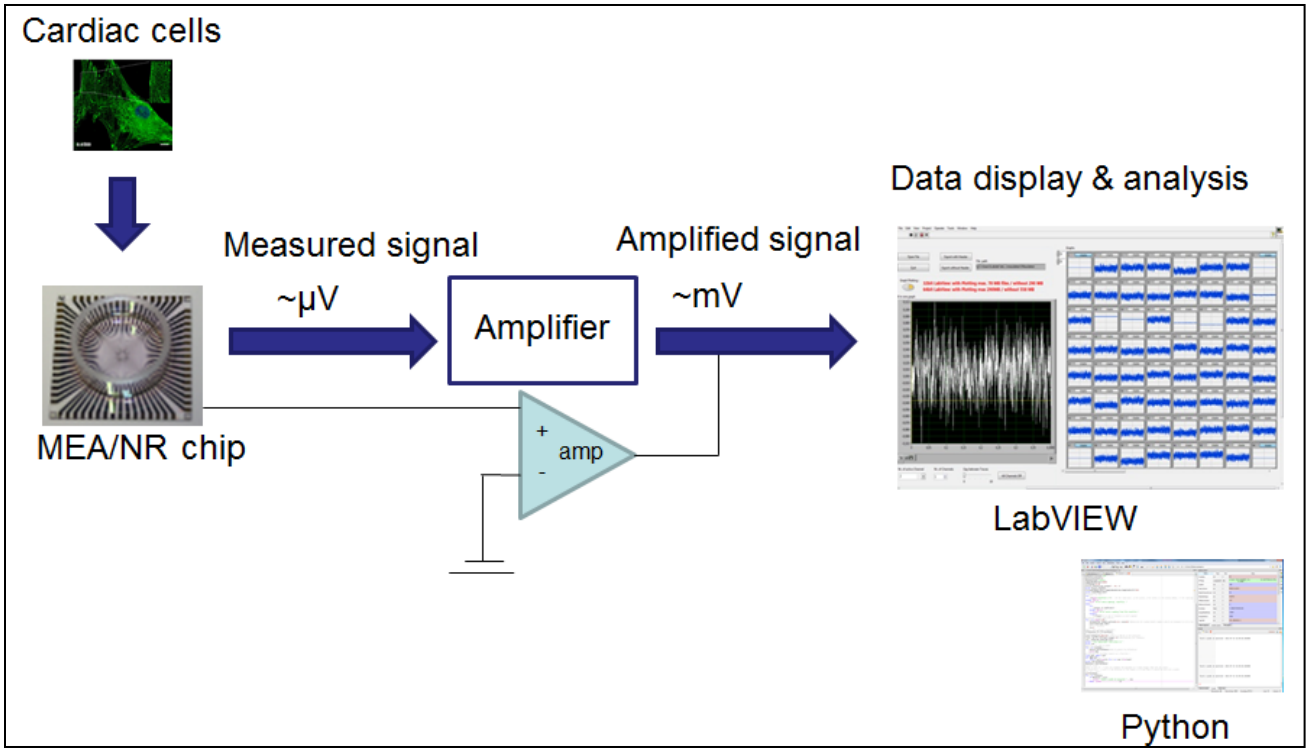



**Figure 17:** Schematic of extra-cellular recordings and data analysis.

In the measurement setup (Figure 18), the amplifier plays an important role in recording the cell activity. The amplifier consists of two parts, the pre-amplifier, and the main amplifier. There are different kinds of head stages, which can be used for different types of chips, for example, they can be used for MEA or Si-FET chips.

The main amplifier has a selectable gain of 1x, 10x or 100x, while the exchangeable MEA preamplifier ("MEA3") has a gain of 10x, resulting therefore, in a maximum total gain of 1000. In addition to a DC-coupled operation mode, the amplifier is also equipped with a selectable AC high-pass filter ranging from 0.01 Hz to 72 Hz (cut-off frequency) and limits the high-frequency regime with a cut-off filter to 3 kHz (at a main amp gain of 100x). The amplifier has a front and a back panel (Figure 19 and Figure 20).

In the recording process, a data acquisition (DAQ) card (USB-6255, National Instruments) was used to sample the output voltage of all 64/30 channels. The next step was to display the signals in the channels with LabVIEW. This was performed with the viewer program, as will be shown in chapter 4.

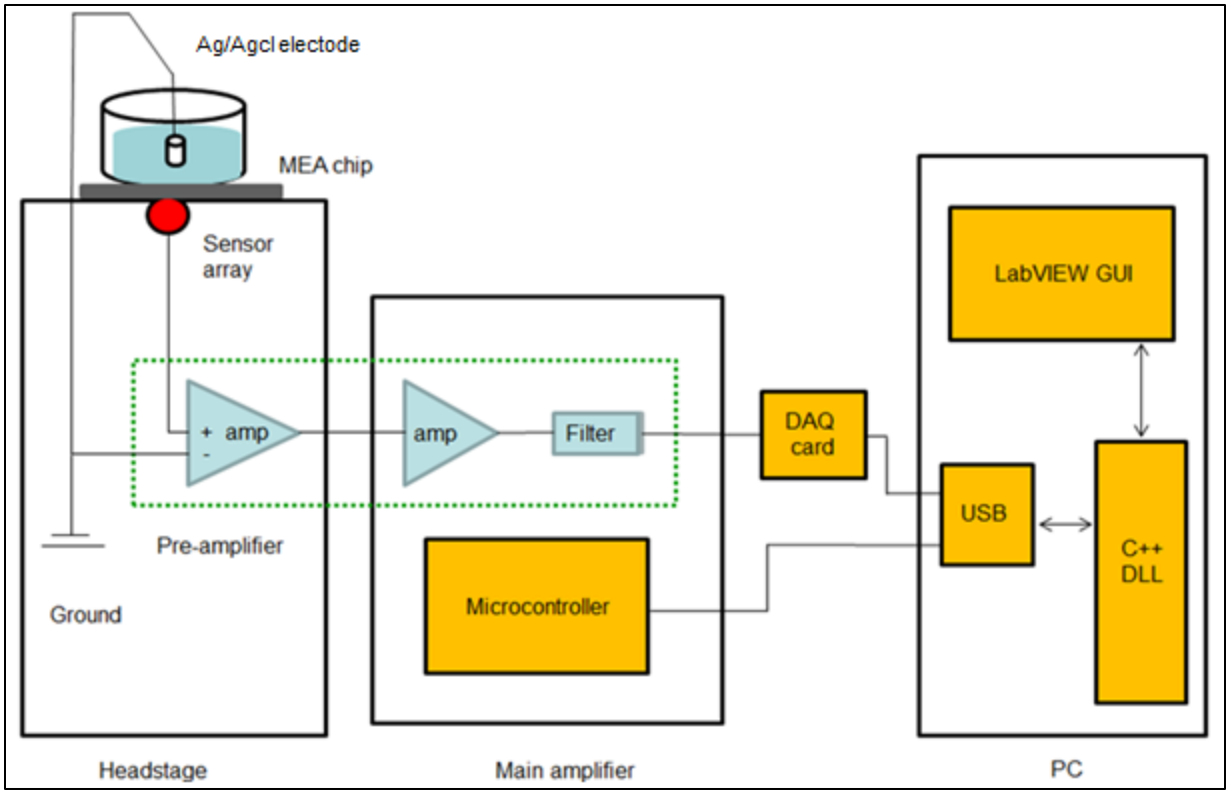



**Figure 18:** Schematic of the electronic readout and control of the amplifier setup.

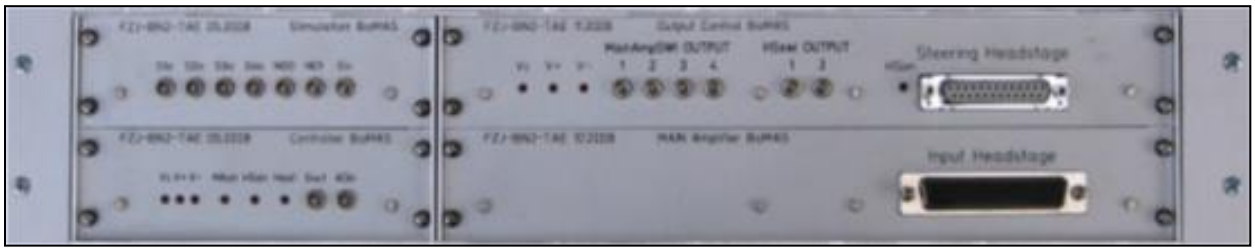


**Figure 19:** Photograph of a front panel (a) [8].

**Figure 20:** Photograph of a back panel (b) [8].

## 3.2 Materials

Cardio myocyte-like cells, a type of cells associated with muscular tissue [16], (Figure 21) were used in the experiment.

There are different kinds and forms of myocytes, such as cardiac, skeletal and smooth muscle cells. Most of these cells generate electrical impulses, which control the heart rate.

The HL-1 cell line (Louisiana State University Health Science Center, New Orleans, USA), was derived from AT-1 cells (a mouse cardiac myocyte tumor). The HL-1 cell line (cardiac cells) was chosen because it is easily adapted to live in a cell culture [17]. These cells are characterized by generating stable, spontaneous AP in regular frequencies [18]. A cell culture (see Appendix C) kept the cells at a suitable temperature (37 $^{o}$C), in saturated water, and fed with the necessary nutrition. The cells were then put in a suitable medium on the MEA chip.

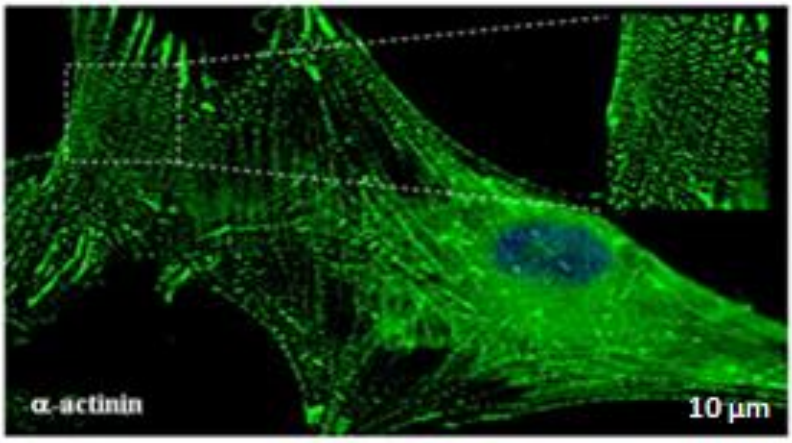



**Figure 21:** An example of a cardio myocyte-like cell [19].

# 4. Results and Discussion

## 4.1 The Viewer

The viewer program is in LabVIEW, which enables the acquired data from a Multi-Electrode Array (MEA) chip to be displayed. The individual channels (Figure 22) are displayed in the "graphs" chart to the right of the main program. The graphs chart consists of 64 small graphs (array of charts) from channel 1 to channel 64. Each chart has an individual button to disable the channel manually if the signal is not of interest. For the MEA, all 64 channels were displayed.

On the left, all channels are displayed in an "all in one graph" chart, which can display all the signals at the same time on a larger scale. Additionally, many options have been modified (see Appendix A) for the user in order to choose a single channel in this graph. Therefore, the user is able to choose between displaying all channels in one graph or even to select each active or measured channel separately in the "all in one graph" chart. It is possible to zoom in the graph at specific time intervals. The x-axis shows time (in seconds) and the y-axis the voltage output (in volts).

Other functions were also introduced. For example, it is possible to add additional channels to the desired single channel in order to compare them. The gaps between the signals could be also varied while running the program with the "gap between traces" button.

The buttons on the upper left side enable the user to import the file through the "open file" button. The "quit" button stops the program. Two functions have been modified to export the data with or without header. These functions export the data into column-based of ASCII data functions, which writes the data to an output file for the evaluation and other uses, and which can be opened using any text program. Next to these buttons, a path indicator was built to show the file destination. The maximum file size that can be imported, is written (in red) as a notification for the user.

The orientation of the active channels was taken into consideration, as will be discussed in more detail in **chapter 4.1.1**. There is also a button to enable/disable the charts so that the user has the opportunity to export the data without enabling the graphs.

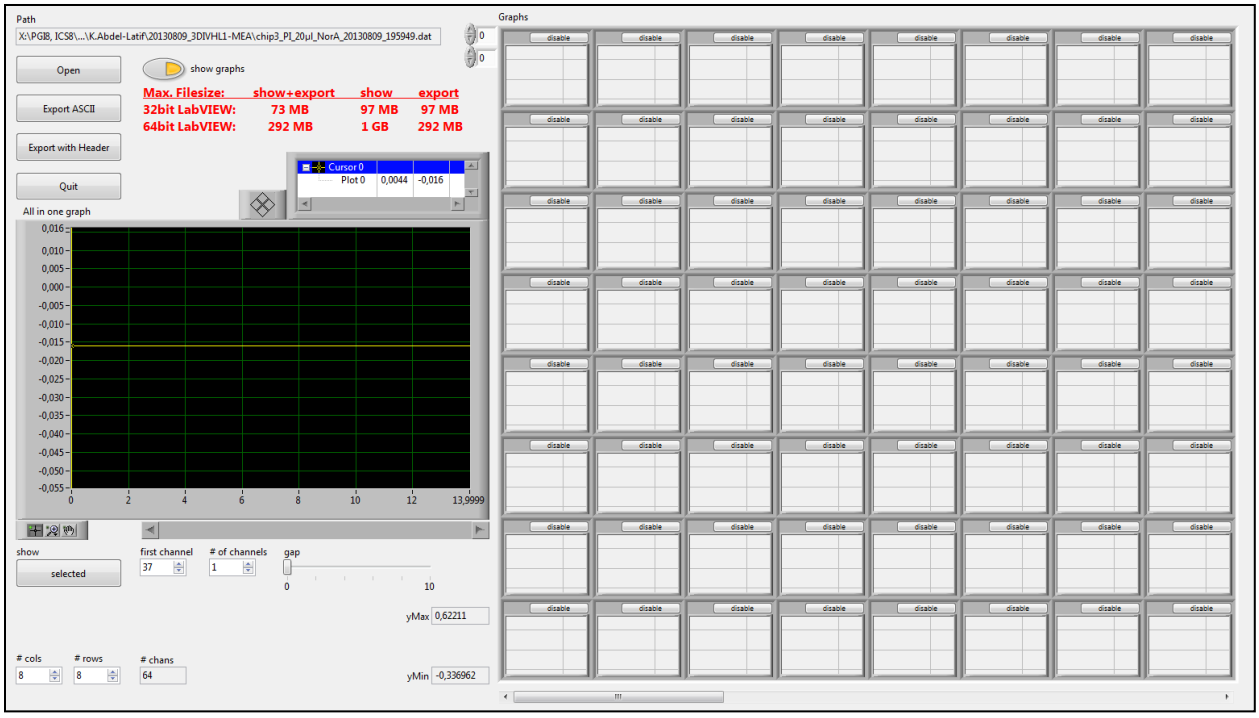



**Figure 22:** Overview of the viewer program.

The "BioMAS" software (the Viewer) is a tool developed for displaying and analyzing the amplified cellular signals. During this thesis, several optimizations to increase the functionality of this tool were performed:

- Displaying the missing channels.
- Ordering the channels in the right arrangement.
- Enabling the manual disabling of the individual channels.
- Illustrating the maximum/minimum values of the individual channels.
- Enabling/disabling the graphs.
- Optimizing the efficiency of data handling.
- Optimization of the exporting function.
- Selecting the currently active channels.

In the following sections, the individual tasks to optimize the display of the signals are described. Furthermore, the path of addressing the problems and the results will be discussed.

### 4.1.1 Displaying Missing Channels

One of the functions of the viewer is to plot each channel individually in the "graphs" charts. Problematic was that the last channel 64 was missing in the lower right corner of the chart (Figure 23).

Channel 64 has the index number 63. The index number of an array differs from the channel number; the index ranges from 0 to 63 but the channel numbers are from 1 to 64.

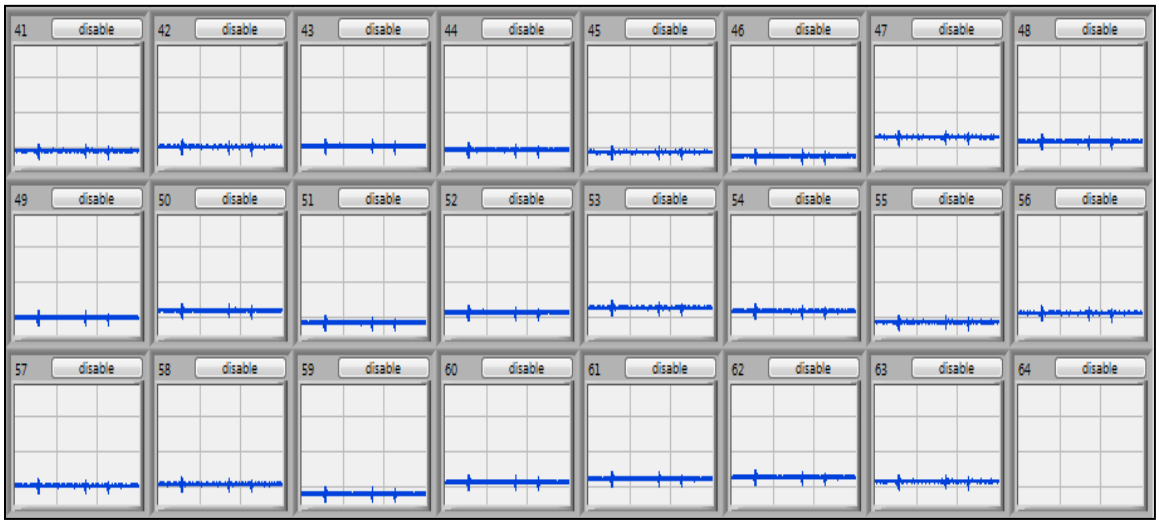



**Figure 23:** Last channel 64 in the lower right corner was missing.

To solve this problem, the "read from file SubVI" command in the main program was changed. The "SubVI" is the orange box (Figure 24). The "SubVI" is an inner programming part of the main program and is equivalent to "functions" in other programming languages. It can be be used in other Virtual Instruments (VI). "SubVIs" reduce code, are easier to debug and give a clear arrangement.

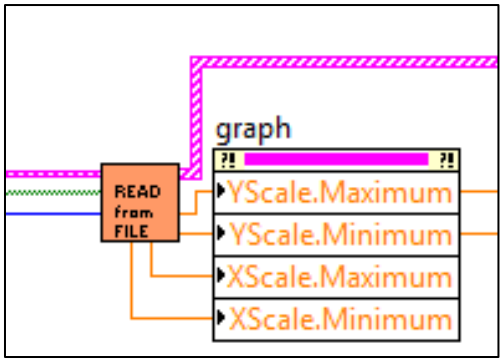



**Figure 24:** "Read from file SubVI".

In the block diagram of "read from file SubVI," the active channels array string (starting from 1 to 64) is compared with the iteration number "i" (the iteration of the loop) which starts from 0 to 63.

First, a decrement by one is added after the index array of the active channels so that every channel will be compared with the right correspondent iteration index of "i" (Figure 25).

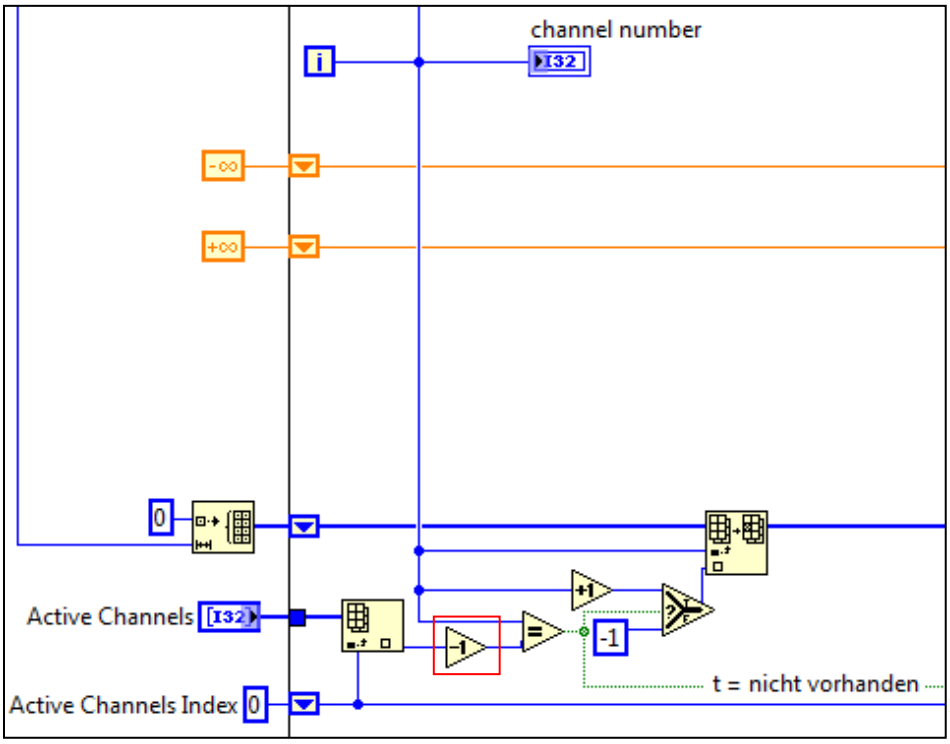



**Figure 25:** Comparison between active channels and iteration number.

Second, for the orientation of the channels, the index array (Figure 26) is changed (in red) to be before the comparison (in green) between the active channel index and the "Boolean" variable of the non-measured channels. In comparison, if the channel is active then the index will be incremented with one. If the channel is not available, the number stays without being incremented. In this case, the index array should be initialized with channel 1 with index 0 and then increments.

Therefore, the comparison was shifted to the right of the index array. In this update, channel 1 was initialized in the index array. After this process, the last channel 64 was displayed (Figure 27).

Moreover, an additional function was created to evaluate the maximum and the minimum of the signal in the individual channel. In the main program, the maximum was taken from the property node (Figure 28). During the manual disabling procedure, it was important to have an indicator, which states the position of the currently disabled channel in the "graphs" chart.

So that the disabled channel can be directly recognized. This was built in the event of the "value change of the graphs" (Figure 29).

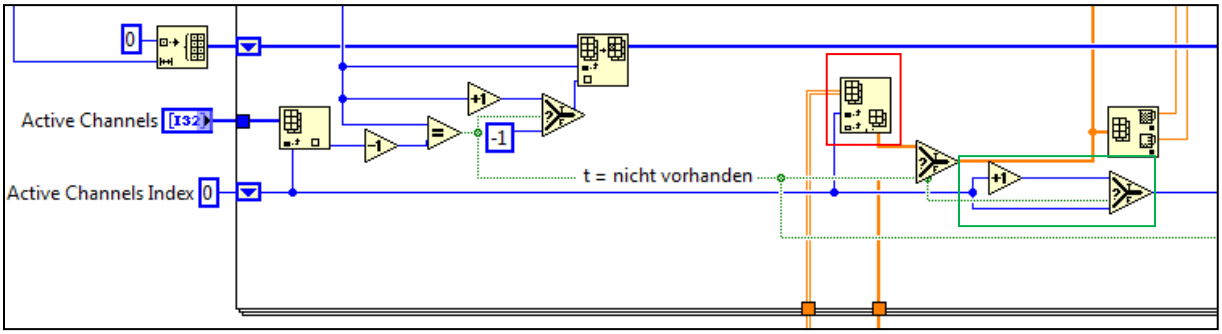



**Figure 26:** Solving the problem of the wrong arrangement of the channels.

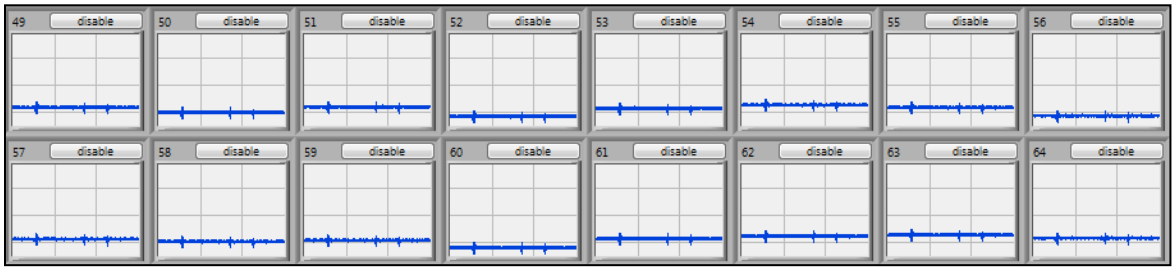



**Figure 27**: Last channel 64 was displayed correctly.

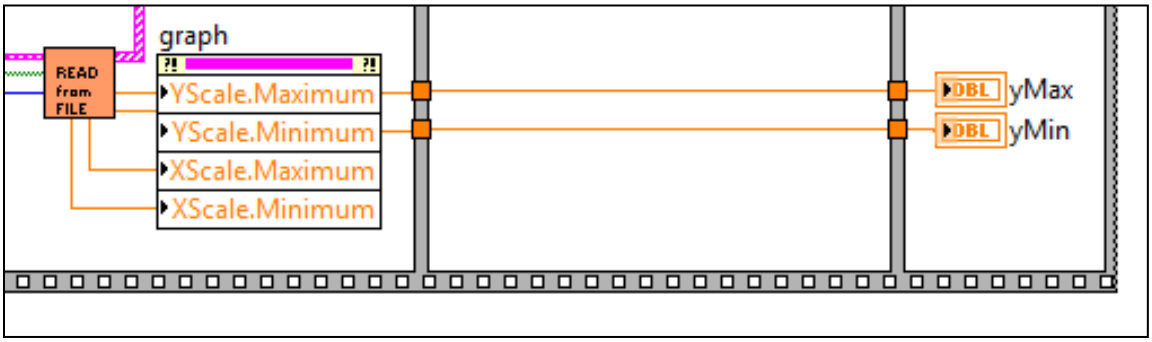



**Figure 28:** Property node for max/min functions.

The old value of the channel and the new value were entered into two "for loops," with an iteration of 8 times (the overall iteration is 8 x 8 = 64 times). The two values were then converted from the string value to a "Boolean" value.

After converting these values, they were compared using a "not-equal Boolean" comparison. If the values were not equal then this was the "True" case (Figure 29). The channel was then disabled and the position of the channel was given from the iterations of the "for loops."

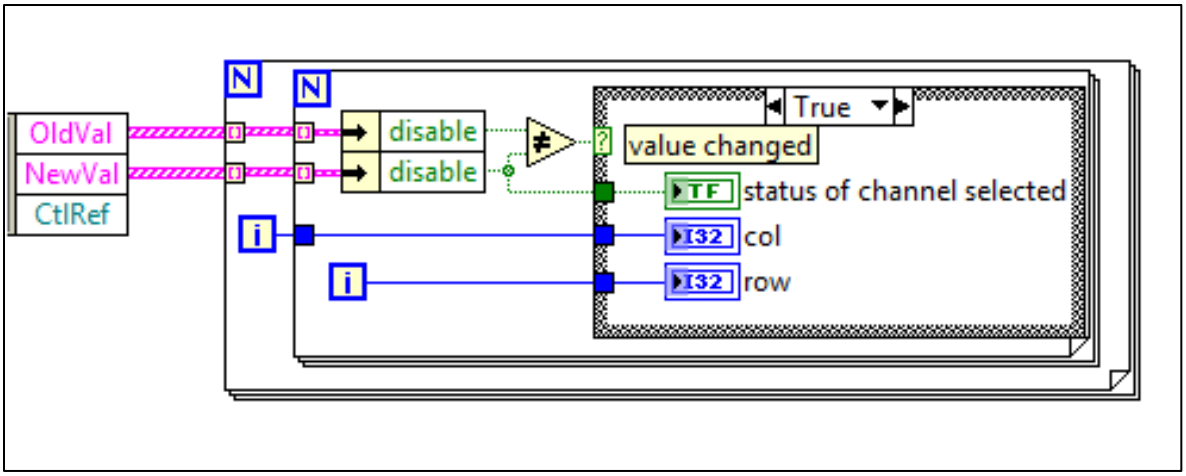



**Figure 29:** "Value change of the graphs" in the disable state (the "True" case).

If the status of the selected channel turned into "True" (the "Boolean" indicator will be on), this showed that a channel was being disabled. The iteration of the outer "for loop" gave the number of column and the iteration of the inner loop gave the corresponding row. In the "False" case (when the old and the new value were not changed), the indicator will be turned off and no change will be shown (Figure 30).

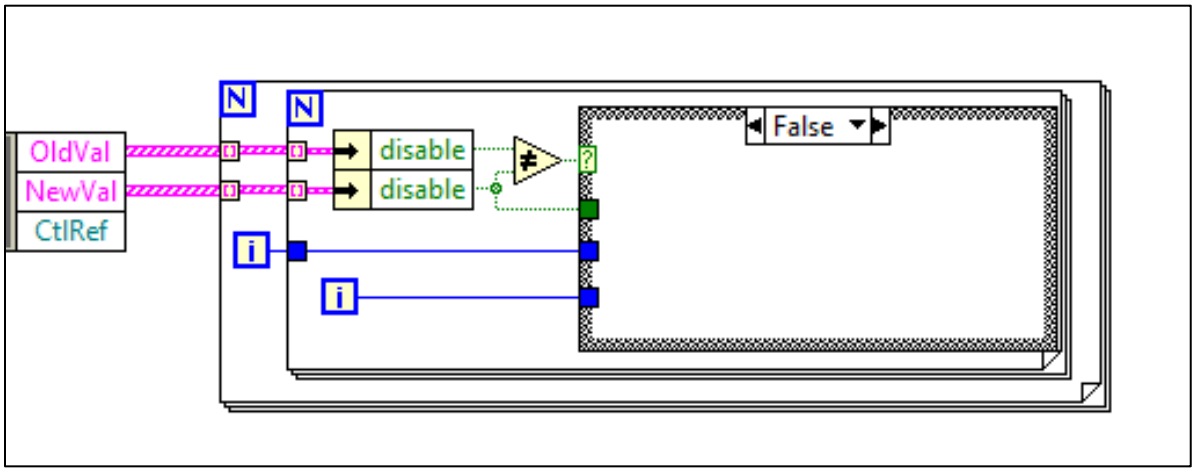



**Figure 30:** The "False" case when the indicator is turned off.

Figure 31 shows the display when, for example, channel 29 was disabled manually. The indicator shows that the disabled channel was in the fourth row and the fifth column. It was taken into consideration that the first row and column had the number zero, as the corresponding index was zero.

The "graphs" chart was shown as a matrix and the single graphs as the element components of the matrix.

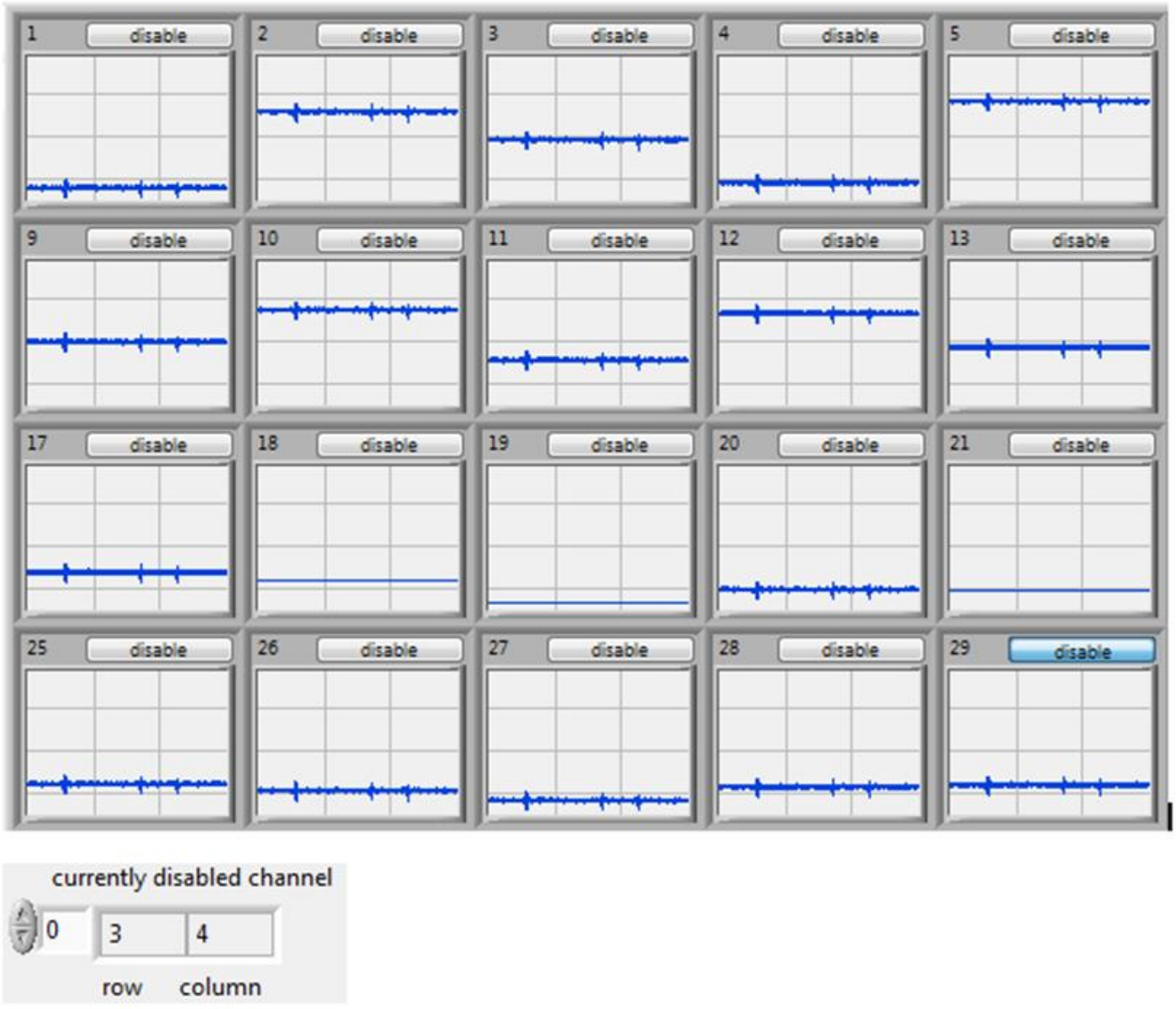



**Figure 31:** An example of disabling the channel 29 in the front panel.

### 4.1.2 Efficient Data Handling

It is important to save as much memory as possible. Therefore, a button has been built to enable or disable the charts. To make this function, a local variable or property node from the charts was put inside a case structure with two cases. The first was the "True" case where the charts were enabled and the second was the "False" case, where an empty array was connected with a wire to the charts so that they were empty and could save up to, for example, 3306 MB of the memory usage while running an input file of size 292 MB (will be discussed in more detail in **chapter 4.1.3**). The "True" case and the "False" case are displayed in Figure 32 and Figure 33.

The same procedure was implemented with the functions in the "all in one graph" chart.

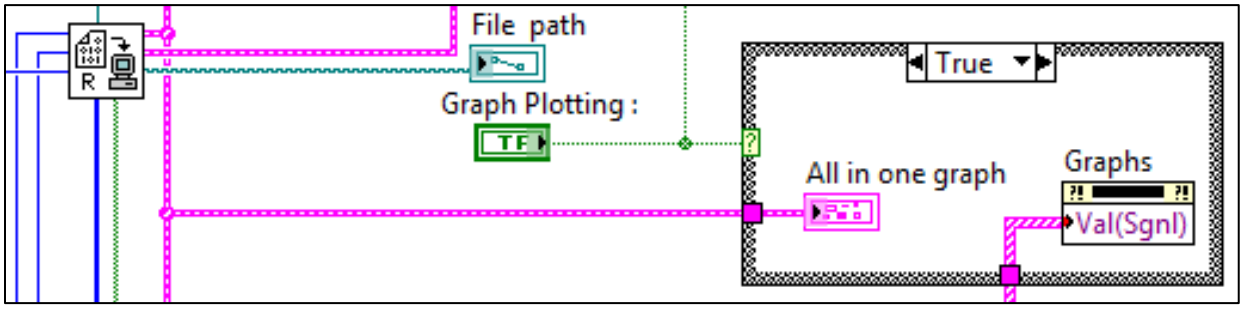



**Figure 32:** Enabling the graphs in the "True" case.

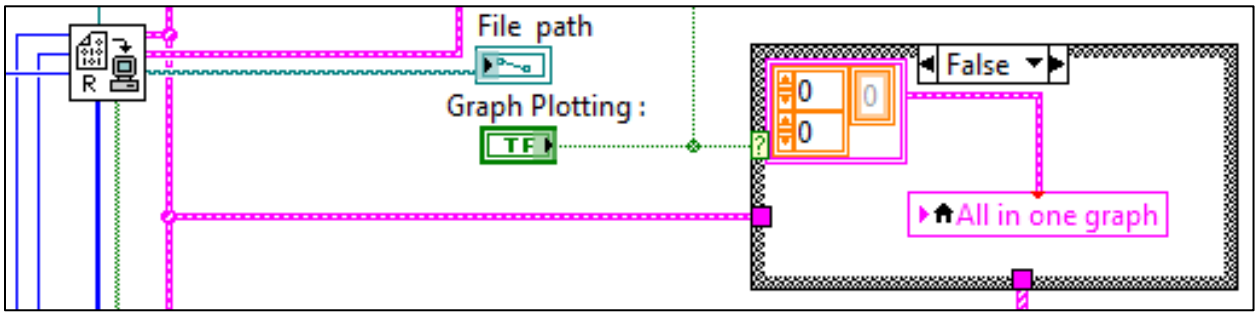



**Figure 33:** Disabling the graphs in the "False" case.

Figure 34 shows the enabling of the individual functions and Figure 35 shows the disabling. An example is also shown of how the button worked while running the program in the front panel. Figure 36 and Figure 37 display how the graphs were disabled and enabled.

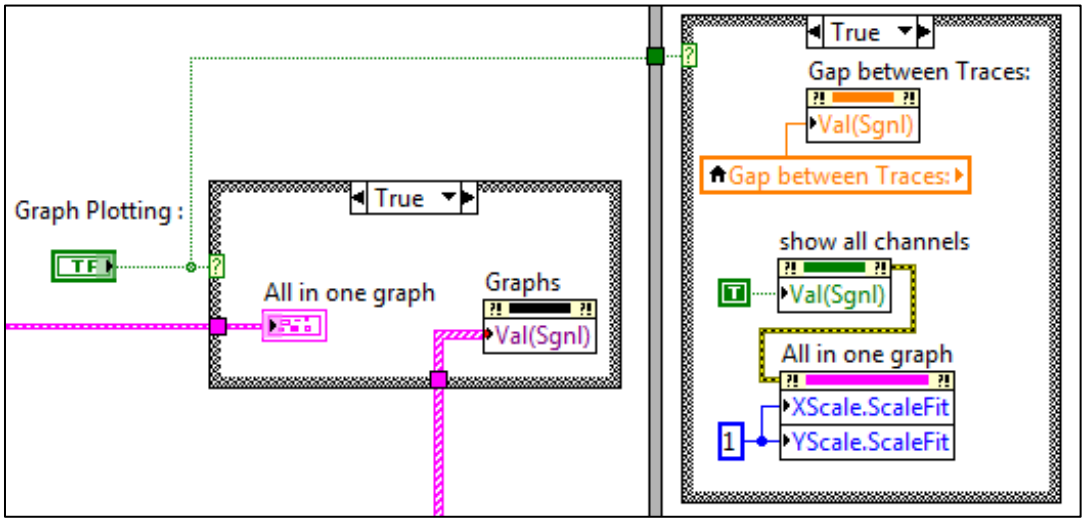



**Figure 34:** Enabling the functions in the "all in one graph" chart.

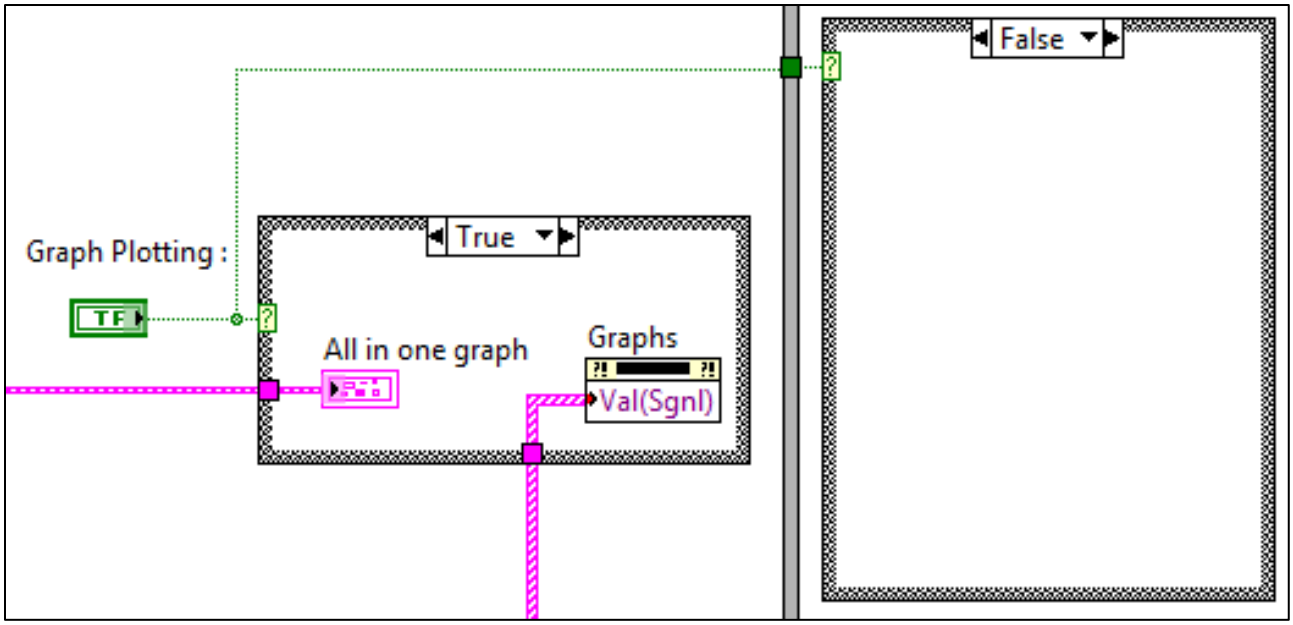



**Figure 35:** Disabling the functions in the "all in one graph" chart.

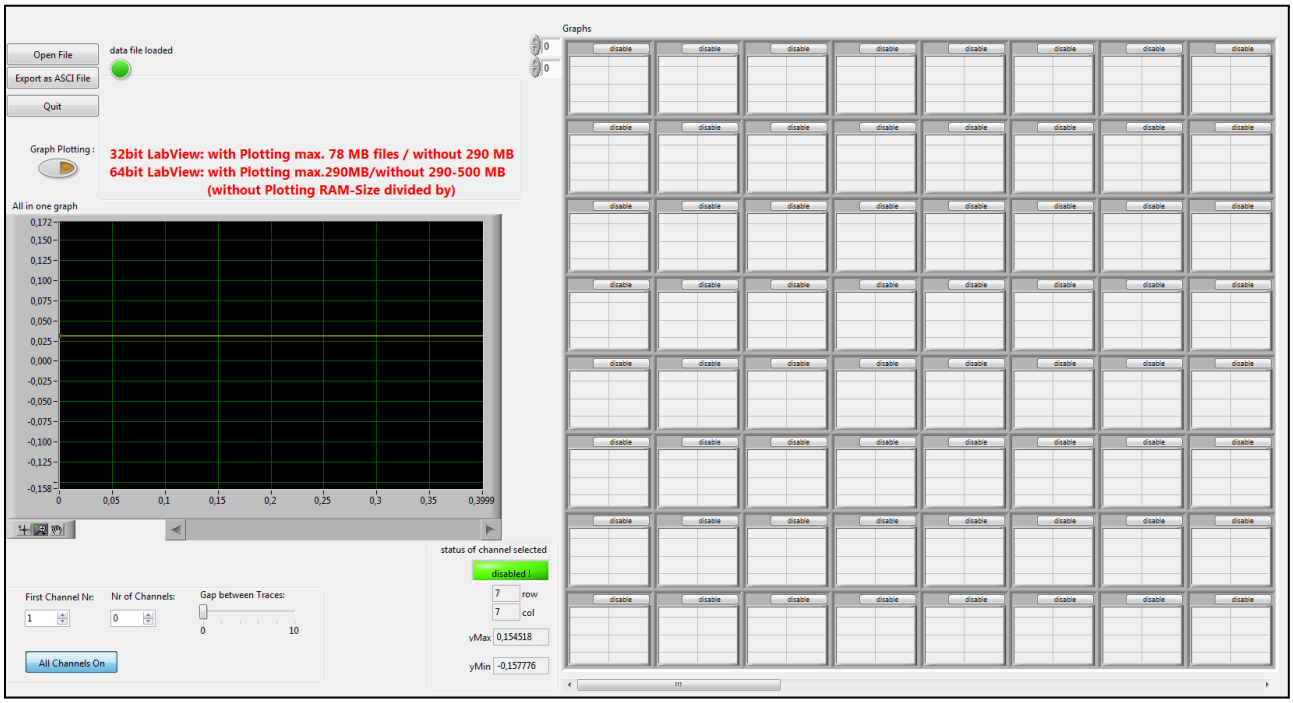



**Figure 36:** Charts disabled while running the program.

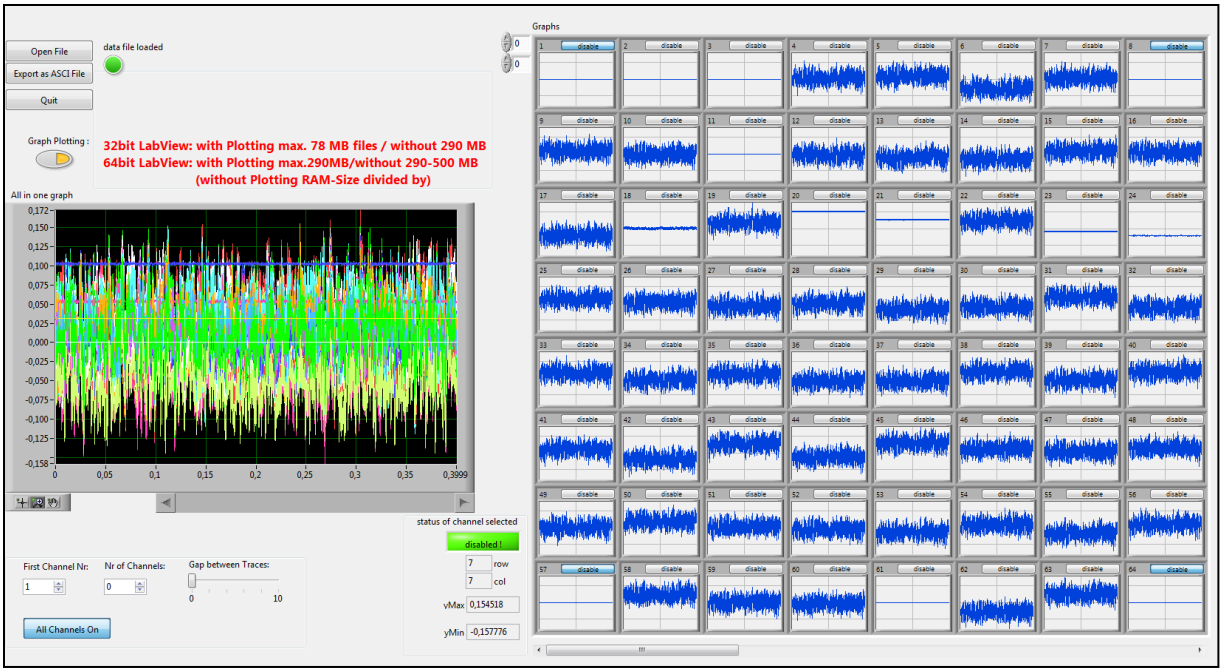



**Figure 37:** Charts enabled while running the program.

Another procedure to save memory is shown in Figure 38 and Figure 39. The "add disabled channels SubVI" (which adds the disabled channels to the data from file to have finally all the channels as a result) and "read file to graph SubVI" (that displays all the channels into the "charts" graph) used the same procedure to fill zeros to the disabled channels, which also used unnecessarily large amounts of memory.

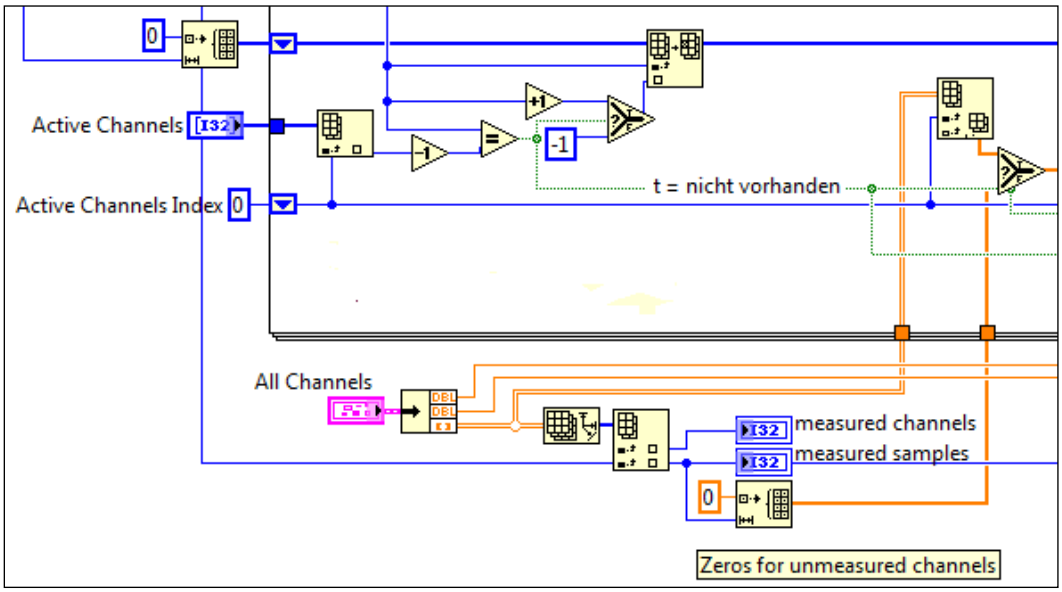



**Figure 38:** "Add disabled channels SubVI".

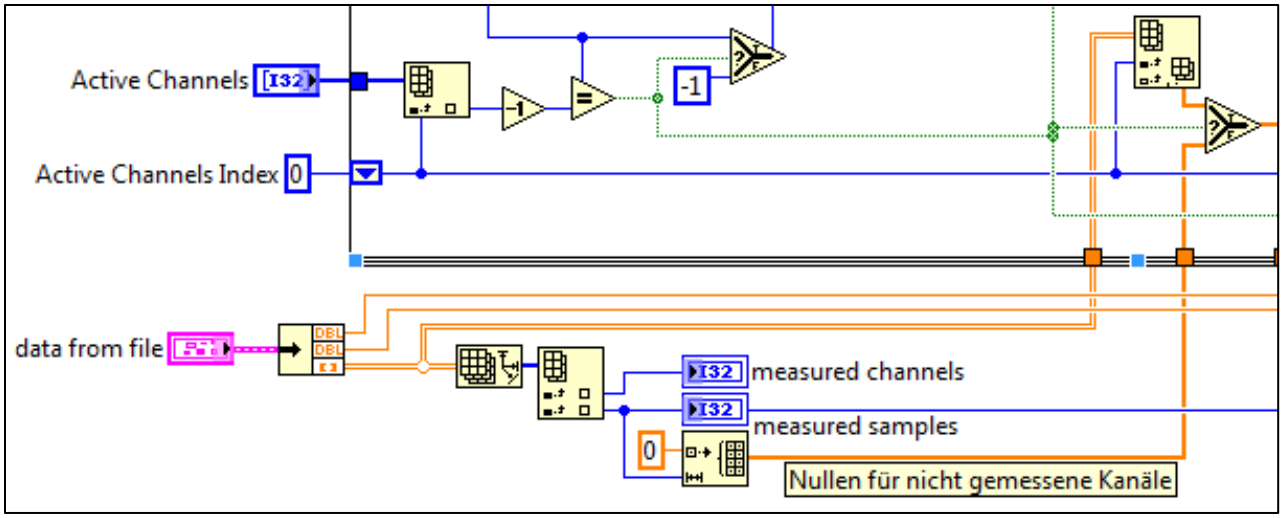



**Figure 39:** "Read from file to graph SubVI".

In order to address this problem, an indicator (Figure 40) was created with the name "non-measured channels" in the "add disabled channels SubVI" with the input of the "Boolean" signal, which indicated whether the channel was disabled or not. This indicator was given as an output of the "add unmeasured channels SubVI" (Figure 41).

After that, this was taken as an output from the "read binary file SubVI" and connected (Figure 42) to the "read from file to graph SubVI" as an input. The new structure of the "read from file to graph SubVI" is shown in Figure 43. As a result, there were no zeros to fill the disabled channels in the "read files to graph SubVI".

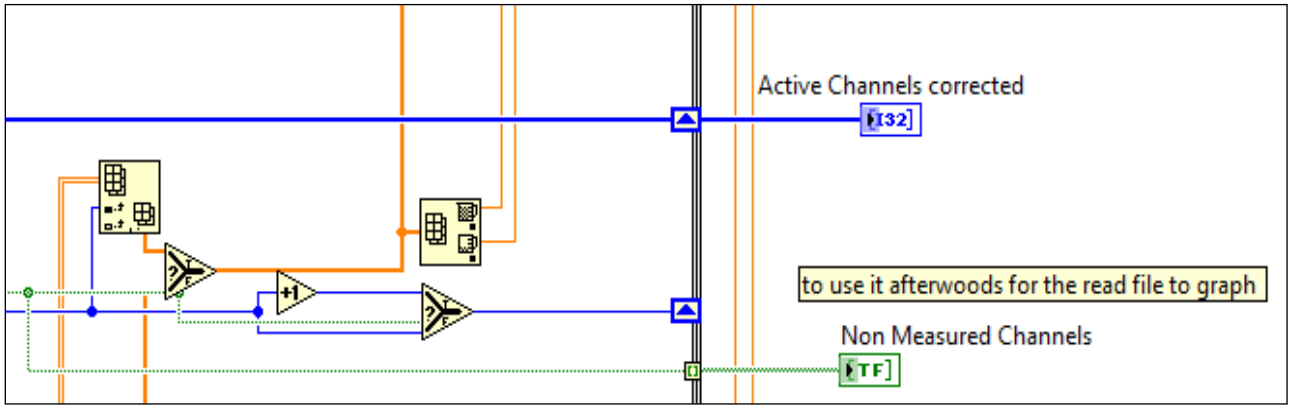



**Figure 40:** Non-measured channel indicator (in green).

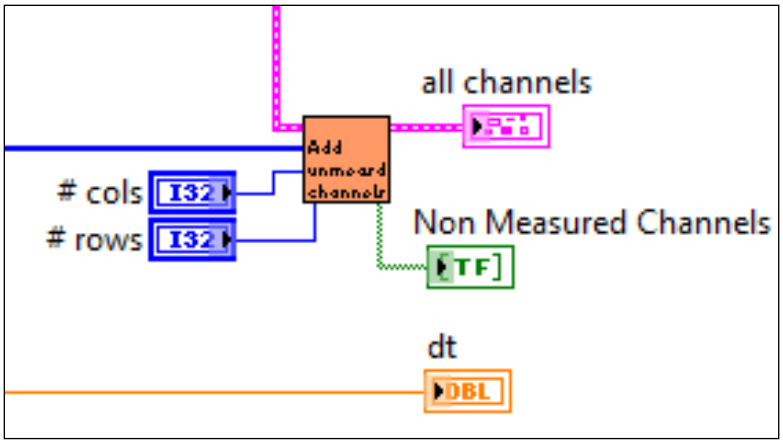



**Figure 41:** Taking the non-measured channel indicator as an output of the "add non-measured channels SubVI".

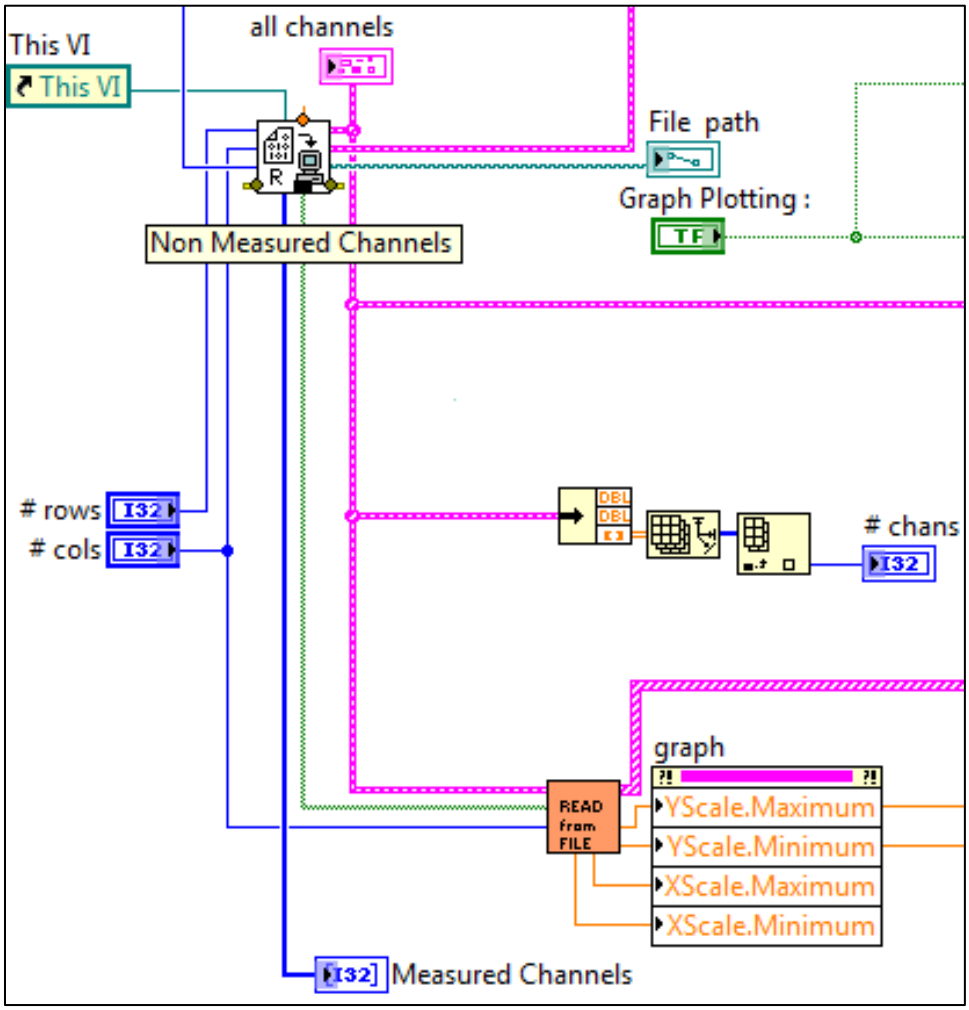



**Figure 42:** Connecting the signal from "read binary file SubVI" to the "read from file to graph SubVI".

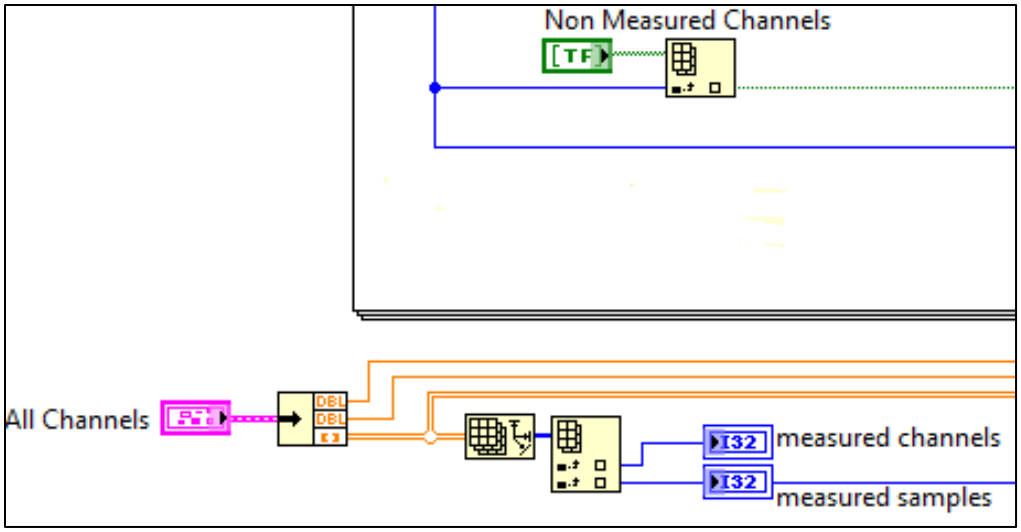



**Figure 43:** The new structure of the "read from file to graph SubVI".

### 4.1.3 Data Export

Exporting the data is very important as it offers the saving of the data. Additionally, it gives information about how many channels are activated as well as displaying the right orientation of the channels within the corresponding time intervals. The user can get the information even while disabling the graphs.

A further aim was to establish a header for the export file column to display the time and corresponding channel number for later analysis. The user has the option of exporting with or without the headers. Decimal points instead of commas are used in the exported file so that it would be compatible with the software written in "Matlab". A path to the file in the main program was created.

The initial state of exporting the data is illustrated in Figure 44. At that time, the orientation of the channels was wrong, as the channels listed were not the active channels. The input to the "unbundle" function was the "all channels" local variable.

After developing the structure, the input data illustrated in Figure 45 was changed to the "data from file" Figure 45. Therefore, at that point, the active channels are the only representatives and there are no additional channels. "Write to spreadsheet SubVI" was changed from %.5f to "%.;%f" to change from commas to decimal points. The next step was the creation of the headers in the exported file using several procedures.

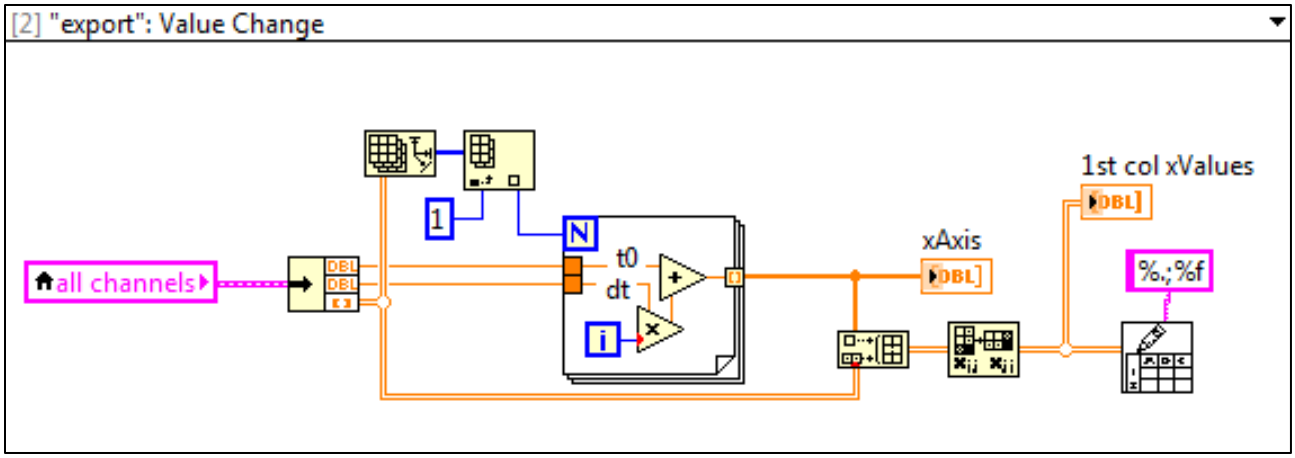



**Figure 44:** Export function in the older initial version.

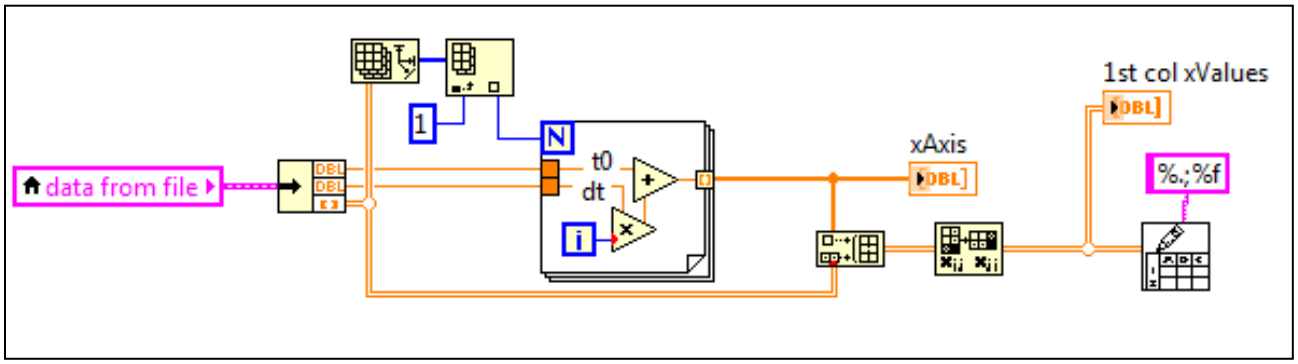



**Figure 45:** Updated export functions without headers.

First, another loop was built (Figure 46) to iterate with the corresponding number of the active channel count. Second, the measured channels were connected to an inverter, which inverts them to a decimal string and then to an index array that is combined with a string-named "Channel" for the title of each channel.

After the loop, these two concatenated strings were connected to the "build array" function, which had a string named "Time [sec]". This was the title of the first column in the export file (the header of the time column). After the build array, the output was connected to a special "write to a spreadsheet SubVI," which writes the information to the output file using the desired headers as string input.

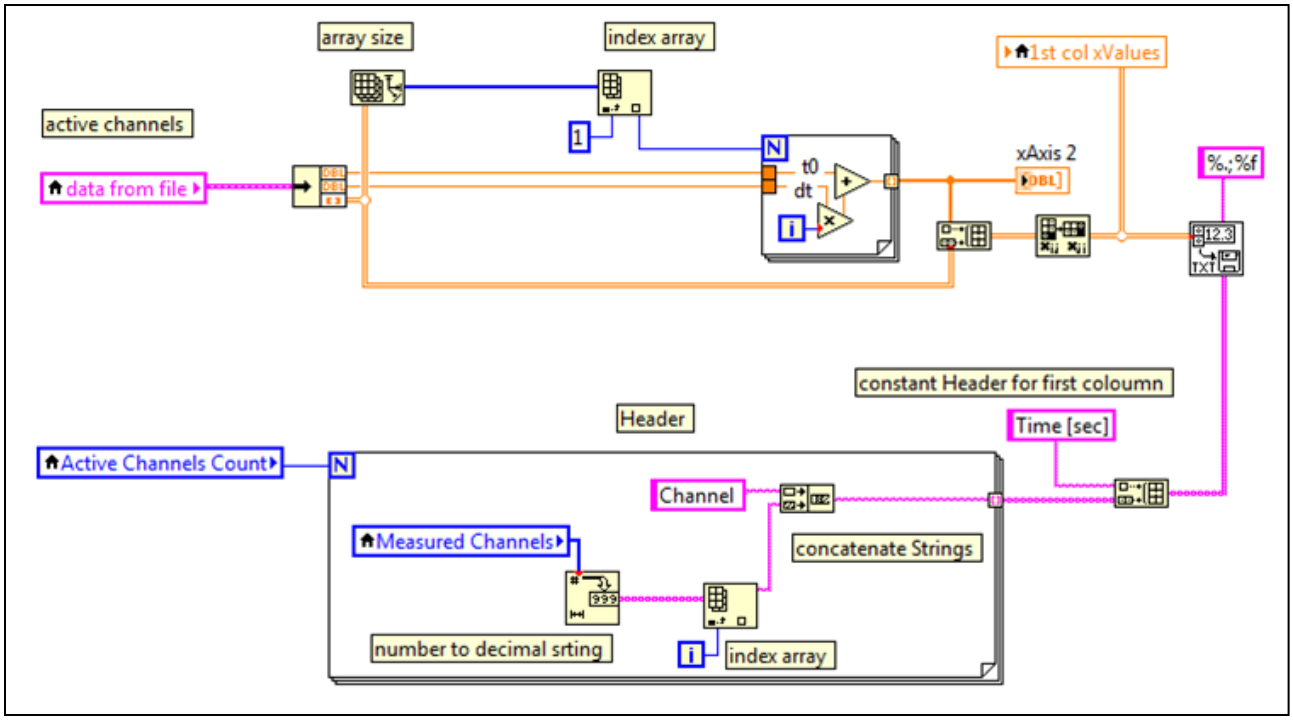



**Figure 46:** The structures of the export function with headers.

This is an example of a testing file where channel 8 is disabled: The file was exported in the older version of the exporting function (Figure 47). It is clear that channel 8 (which is disabled) was displayed (in red) and filled with zeros. This uses extra memory space.

File Edit Format View Help

| | | | | |
|---|---|---|---|---|
| 6,750431 | 0,000000 | 2,504459 | 4,400776 | 1,406917 |
| 6,751397 | 0,000000 | 2,502849 | 4,390152 | 1,404986 |
| 6,748499 | 0,000000 | 2,500273 | 4,385001 | 1,396615 |
| 6,768461 | 0,000000 | 2,516049 | 4,399488 | 1,418508 |
| 6,786169 | 0,000000 | 2,535688 | 4,422348 | 1,437825 |
| 6,772325 | 0,000000 | 2,519912 | 4,412689 | 1,438791 |
| 6,760734 | 0,000000 | 2,518625 | 4,399167 | 1,419796 |
| 6,745280 | 0,000000 | 2,495444 | 4,372766 | 1,394683 |
| 6,745280 | 0,000000 | 2,499951 | 4,379527 | 1,405308 |
| 6,760412 | 0,000000 | 2,508322 | 4,397235 | 1,418186 |
| 6,747533 | 0,000000 | 2,496088 | 4,386288 | 1,409815 |
| 6,748499 | 0,000000 | 2,486107 | 4,377595 | 1,403054 |
| 6,751719 | 0,000000 | 2,490293 | 4,382103 | 1,402410 |
| 6,760734 | 0,000000 | 2,507356 | 4,390152 | 1,406917 |
| 6,758480 | 0,000000 | 2,501883 | 4,386288 | 1,409493 |

**Figure 47:** Exported file (channel 8 disabled) in the older version.

After adding the developed export function, this problem was solved (Figure 48). The changes are clearer in the "exporting with headers" function.

In Figure 49, the headers are above each corresponding channel and only the activated channels are displayed. Note that channel 8 is not displayed, which means that it is disabled. This eliminates unnecessarily filling the disabled channels with zeros.

| | | | | |
|---|---|---|---|---|
| 6.750431 | 2.504459 | 4.400776 | 1.406917 | 4.184100 |
| 6.751397 | 2.502849 | 4.390152 | 1.404986 | 4.175729 |
| 6.748499 | 2.500273 | 4.385001 | 1.396615 | 4.163816 |
| 6.768461 | 2.516049 | 4.399488 | 1.418508 | 4.191827 |
| 6.786169 | 2.535688 | 4.422348 | 1.437825 | 4.211466 |
| 6.772325 | 2.519912 | 4.412689 | 1.438791 | 4.199232 |
| 6.760734 | 2.518625 | 4.399167 | 1.419796 | 4.193437 |
| 6.745280 | 2.495444 | 4.372766 | 1.394683 | 4.171544 |
| 6.745280 | 2.499951 | 4.379527 | 1.405308 | 4.174441 |
| 6.760412 | 2.508322 | 4.397235 | 1.418186 | 4.192471 |
| 6.747533 | 2.496088 | 4.386288 | 1.409815 | 4.176051 |
| 6.748499 | 2.486107 | 4.377595 | 1.403054 | 4.164138 |
| 6.751719 | 2.490293 | 4.382103 | 1.402410 | 4.174763 |
| 6.760734 | 2.507356 | 4.390152 | 1.406917 | 4.186997 |
| 6.758480 | 2.501883 | 4.386288 | 1.409493 | 4.183778 |
| 6.776510 | 2.523776 | 4.404318 | 1.430742 | 4.204705 |
| 6.785847 | 2.531825 | 4.410435 | 1.437825 | 4.208569 |
| 6.764597 | 2.518947 | 4.394981 | 1.426235 | 4.196334 |
| 6.738196 | 2.490293 | 4.373088 | 1.400800 | 4.172509 |
| 6.702459 | 2.457775 | 4.344756 | 1.370859 | 4.140635 |

**Figure 48:** "Export file without headers" after using the updated version of the export function.

| Channel 7 | Channel 9 | Channel 10 | Channel 11 | Channel 12 |
|---|---|---|---|---|
| 6.750431 | 2.504459 | 4.400776 | 1.406917 | 4.184100 |
| 6.751397 | 2.502849 | 4.390152 | 1.404986 | 4.175729 |
| 6.748499 | 2.500273 | 4.385001 | 1.396615 | 4.163816 |
| 6.768461 | 2.516049 | 4.399488 | 1.418508 | 4.191827 |
| 6.786169 | 2.535688 | 4.422348 | 1.437825 | 4.211466 |
| 6.772325 | 2.519912 | 4.412689 | 1.438791 | 4.199232 |
| 6.760734 | 2.518625 | 4.399167 | 1.419796 | 4.193437 |
| 6.745280 | 2.495444 | 4.372766 | 1.394683 | 4.171544 |
| 6.745280 | 2.499951 | 4.379527 | 1.405308 | 4.174441 |
| 6.760412 | 2.508322 | 4.397235 | 1.418186 | 4.192471 |
| 6.747533 | 2.496088 | 4.386288 | 1.409815 | 4.176051 |
| 6.748499 | 2.486107 | 4.377595 | 1.403054 | 4.164138 |
| 6.751719 | 2.490293 | 4.382103 | 1.402410 | 4.174763 |
| 6.760734 | 2.507356 | 4.390152 | 1.406917 | 4.186997 |
| 6.758480 | 2.501883 | 4.386288 | 1.409493 | 4.183778 |
| 6.776510 | 2.523776 | 4.404318 | 1.430742 | 4.204705 |
| 6.785847 | 2.531825 | 4.410435 | 1.437825 | 4.208569 |
| 6.764597 | 2.518947 | 4.394981 | 1.426235 | 4.196334 |
| 6.738196 | 2.490293 | 4.373088 | 1.400800 | 4.172509 |
| 6.702459 | 2.457775 | 4.344756 | 1.370859 | 4.140635 |

**Figure 49:** "Export file with headers" after using the updated version of the export function.

The general overview of the export function with headers is seen in Figure 50. To test the memory usage during the export function, measurements were performed in a 32-bit and 64-bit LabVIEW routine.

| Time [sec] | Channel 2 | Channel 3 | Channel 4 | Channel 5 |
|---|---|---|---|---|
| 0.000000 | 0.003522 | 0.027347 | 0.007708 | -0.046381 |
| 0.000100 | -0.022556 | 0.004488 | -0.017727 | -0.070527 |
| 0.000200 | -0.034146 | -0.005814 | -0.025776 | -0.082439 |
| 0.000300 | -0.043483 | -0.019014 | -0.022556 | -0.072781 |
| 0.000400 | -0.047024 | -0.017727 | -0.029317 | -0.079220 |
| 0.000500 | -0.017083 | 0.005132 | 0.001269 | -0.054429 |
| 0.000600 | -0.001307 | 0.023805 | 0.016400 | -0.041873 |
| 0.000700 | 0.008995 | 0.035074 | 0.027991 | -0.026419 |
| 0.000800 | 0.001269 | 0.034430 | 0.019942 | -0.039620 |
| 0.000900 | -0.012253 | 0.021552 | 0.014147 | -0.047024 |
| 0.001000 | -0.003883 | 0.024449 | 0.016400 | -0.048634 |
| 0.001100 | -0.004205 | 0.019298 | 0.011571 | -0.050244 |
| 0.001200 | -0.014829 | 0.005132 | 0.005776 | -0.054429 |
| 0.001300 | -0.009356 | 0.012537 | 0.017688 | -0.044127 |
| 0.001400 | -0.011932 | 0.008352 | 0.019620 | -0.046702 |
| 0.001500 | -0.018371 | -0.009034 | 0.006420 | -0.059259 |
| 0.001600 | -0.036078 | -0.016117 | -0.008068 | -0.070205 |
| 0.001700 | -0.039620 | -0.014829 | -0.003239 | -0.078254 |
| 0.001800 | -0.054751 | -0.028673 | -0.023844 | -0.093064 |
| 0.001900 | -0.091132 | -0.067629 | -0.056361 | -0.126869 |
| 0.002000 | -0.110449 | -0.083727 | -0.078254 | -0.146830 |
| 0.002100 | -0.099503 | -0.078576 | -0.076644 | -0.138137 |
| 0.002200 | -0.084693 | -0.069883 | -0.061512 | -0.123971 |
| 0.002300 | -0.060868 | -0.049922 | -0.038332 | -0.107552 |
| 0.002400 | -0.049278 | -0.033180 | -0.019336 | -0.088878 |
| 0.002500 | -0.050244 | -0.033180 | -0.016761 | -0.079542 |
| 0.002600 | -0.038654 | -0.023200 | -0.003239 | -0.083727 |
| 0.002700 | -0.032858 | -0.011932 | 0.008674 | -0.079864 |
| 0.002800 | -0.024810 | -0.004527 | 0.008030 | -0.074069 |
| 0.002900 | -0.024488 | -0.008068 | 0.004166 | -0.075034 |
| 0.003000 | -0.012575 | 0.006742 | 0.017366 | -0.065376 |

**Figure 50:** General overview of the headers function.

The terms 32-bit and 64-bit refer to the way the CPU (computer's processor) and operating system handle information. While 32-bit platforms are limited to a maximum amount of 2-4 GB (reference for windows platforms), 64-bit offers the thread sizes up to the full amount of installed random access memory (RAM).

After testing several input files in the 32-bit LabVIEW (Table 1), the "Loading + Plotting", "Loading + Export", and the "Loading + Plotting + Export" were compared. The blank elements in the table indicate that there was no memory space needed to complete the procedure. The memory usage of each function was determined by subtracting the current memory usage by the memory usage used to open the program, which is 139,260 KB.

Figure 51 shows the difference between the components of Table 1. "Loading + Plotting" and "Loading + Export" can occur for an input file of 97 MB. The "Loading + Plotting + Export" can occur in import files up to 73 MB. As shown in the figure there is not enough memory space to import larger files.

**Table 1:** Comparison of memory consumption between "Loading + Plotting", "Loading + Export", and the "Loading + Plotting + Export" of the data in 32-bit LabVIEW program.

| File size [MB] | Loading + Plotting [MB] | Loading + Export [MB] | Loading + Plotting + Export [MB] |
|---|---|---|---|
| 0,352 | 596 | 156 | 478 |
| 0,702 | 621 | 163 | 502 |
| 1,055 | 636 | 169 | 507 |
| 1,407 | 641 | 175 | 660 |
| 24 | 932 | 535 | 1088 |
| 49 | 1889 | 1123 | 2131 |
| 73 | 2824 | 1692 | 3463 |
| 97 | 2873 | 2154 | |
| 121 | | | |
| 219 | | | |
| 292 | | | |
| 732 | | | |
| 1024 | | | |

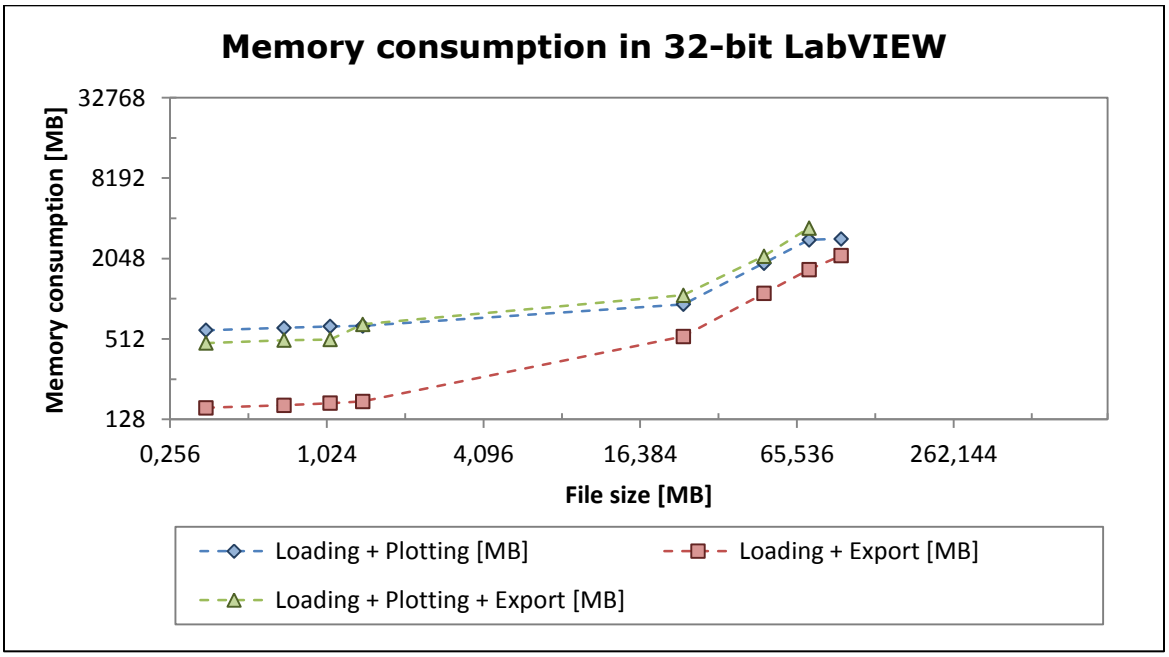



**Figure 51:** Memory consumption of different files sizes in a 32-bit LabVIEW environment.

The same comparison between the three functions is performed in a 64-bit LabVIEW (Table 2) environment. The graph in Figure 52 shows the difference between the measuring procedures in 64-bit LabVIEW program. There is a large difference noticed as the "Loading + Plotting" has been

enabled by files sizes up to 1 GB. The "Loading + Export" and "Loading + Plotting + Export" stopped functioning with file sizes more than 292 MB.

**Table 2:** Comparison of memory consumption between "Loading + Plotting", "Loading + Export", and the "Loading + Plotting + Export" of the data in a 64-bit LabVIEW program.

| File size [MB] | Loading + Plotting [MB] | Loading + Export [MB] | Loading + Plotting + Export [MB] |
|---|---|---|---|
| 0,352 | 618 | 282 | 630 |
| 0,702 | 633 | 300 | 643 |
| 1,055 | 637 | 308 | 651 |
| 1,407 | 642 | 369 | 658 |
| 24 | 934 | 774 | 1140 |
| 49 | 2295 | 1561 | 2296 |
| 73 | 2811 | 1562 | 3451 |
| 97 | 3737 | 3093 | 4589 |
| 121 | 4743 | 3872 | 5752 |
| 219 | 8402 | 6416 | 10326 |
| 292 | 8405 | 6944 | 10330 |
| 732 | 27940 | | |
| 1024 | 30812 | | |

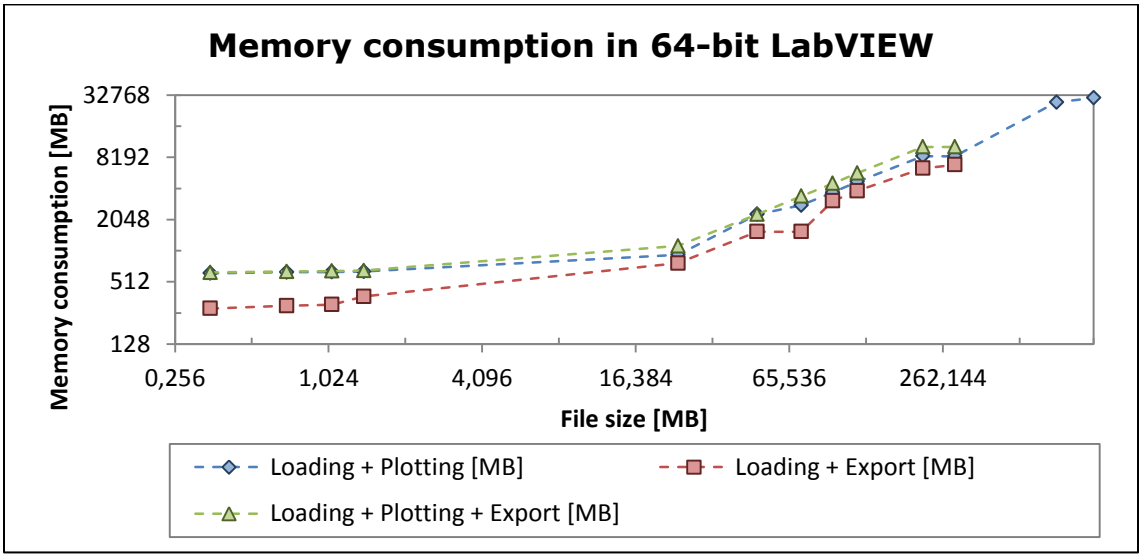



**Figure 52:** Memory consumption for different files sizes in 64-bit LabVIEW environment.

### 4.1.4 Selecting Activated Channels

As discussed in **chapter 4.1**, the "all in one graph" displays all the channels at the same time. A new function has been built up to give the user an option to choose separately the individual, active channels to be displayed

on the graph. Not all the channels were represented in the list, just the active channels, which could then be selected by the user.

To achieve this, a "for loop" has been implemented with the active channel count as iteration (Figure 53). The measured channels were then converted to a string, which was connected to an index array. The output of the "for loop" was then connected to a property node which has the name of "nr. of active channel" (which is the actual number of the activated channel).

The string of the active channel was then given as an array of integers to the array subset. The control list was added to the array subset. This control list is within the combo box where the user chooses the desired channel. The output was connected to the function working on the size of the gaps between the channels.

The Boolean input named "show all channels" is the button "all channels off/on" in the front panel (Figure 54), which enables all the channels in the graph in the "True" case to be shown or only the single channels in the "False" case.

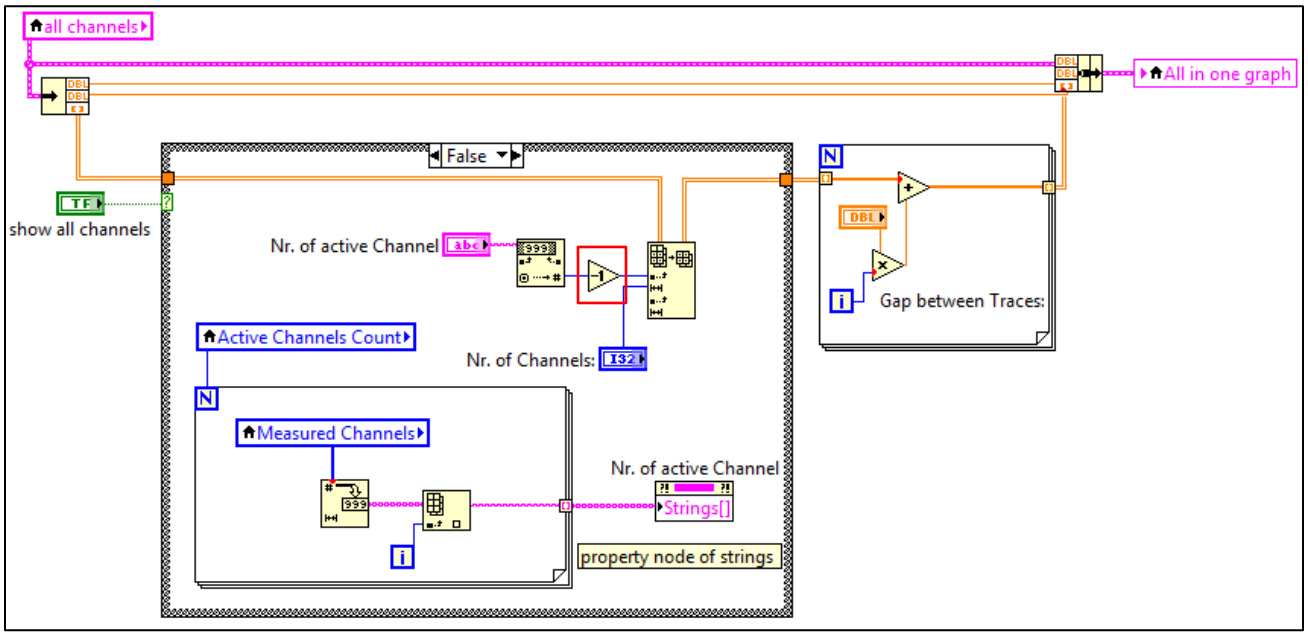



**Figure 53:** Control list for the activated channels.

The decrement (in red) after the decimal-to-string-function was important for the orientation of the channels, as it must be taken into consideration that the first number of active channels has the index of zero, not one.

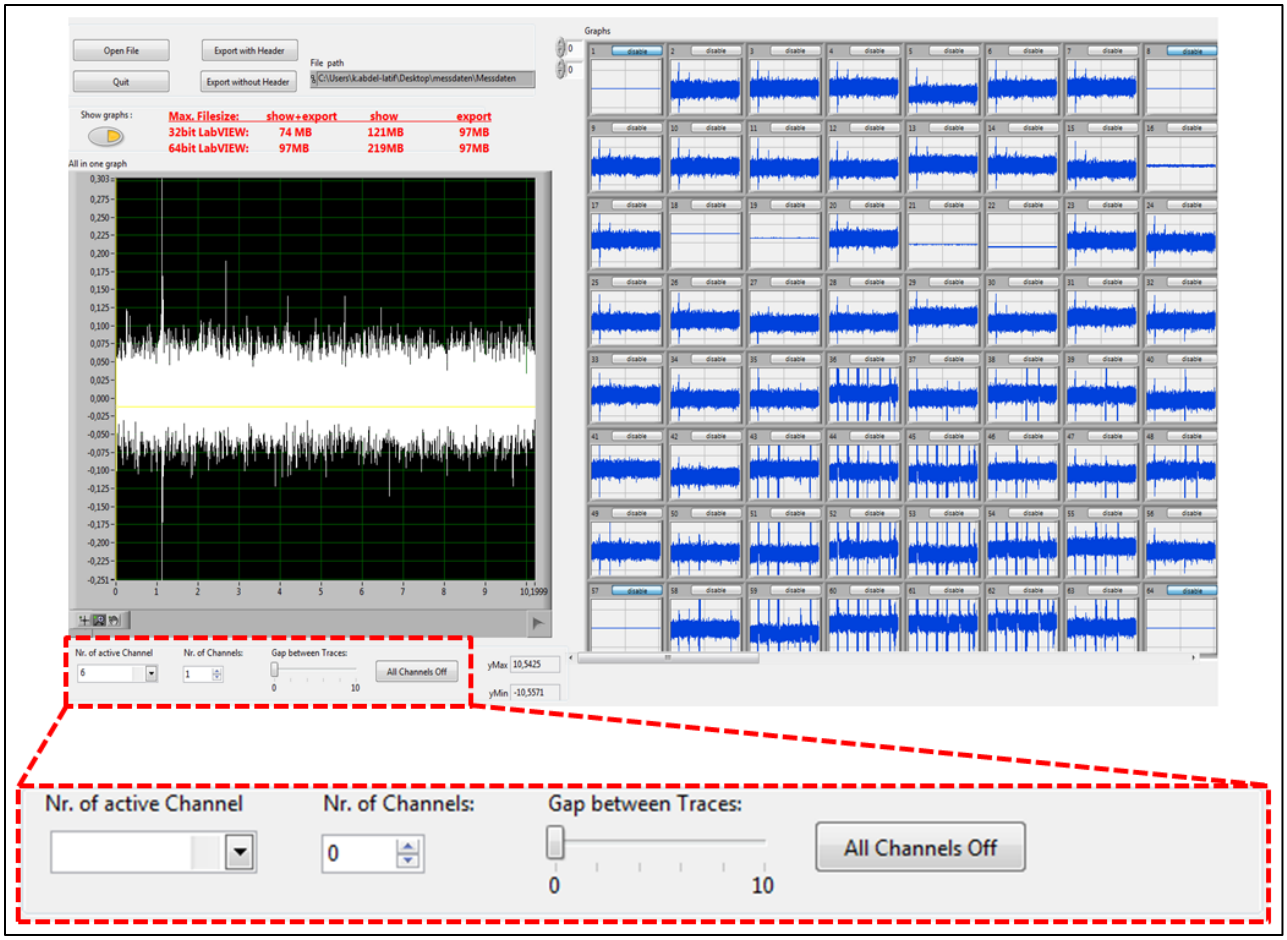



**Figure 54:** Control list and the corresponding buttons while running an input file with disabling the channels 1-8-57-64.

Figure 55 shows an example of the control list on the front panel. Channel 8, which is disabled, does not appear. While running the program, the user can select the desired channels from the combo box. In the "all in one graph" waveform chart, there is a function implemented to add additional channels after selecting the individual channel. The user can easily compare between the individual signals, not just to display one single channel. This was achieved by the control "nr. of Channels" (in blue) in Figure 53.

The output in the chart is shown in this example in the front panel (Figure 56). Channel 3 is shown in white (originally selected in the "nr. of active Channel" control). Channels 4 and 5, which were added in the "nr. of channels" control list, are shown in green and red. In order to make the display and the differences between the channels clearer, the "Gap between Traces" function was introduced.

With this function, the user can slide the desired gap between the channels to get a better overview. An example is shown when the gap is changed on the channels 3, 4 and 5 (Figure 57). For the automatic scaling function, a property node with the name "YScale.ScaleFit" was developed. This allows the user to zoom in to get the full automatic scaling. Figure 58 and Figure 59 illustrate the function before and after auto-scaling.

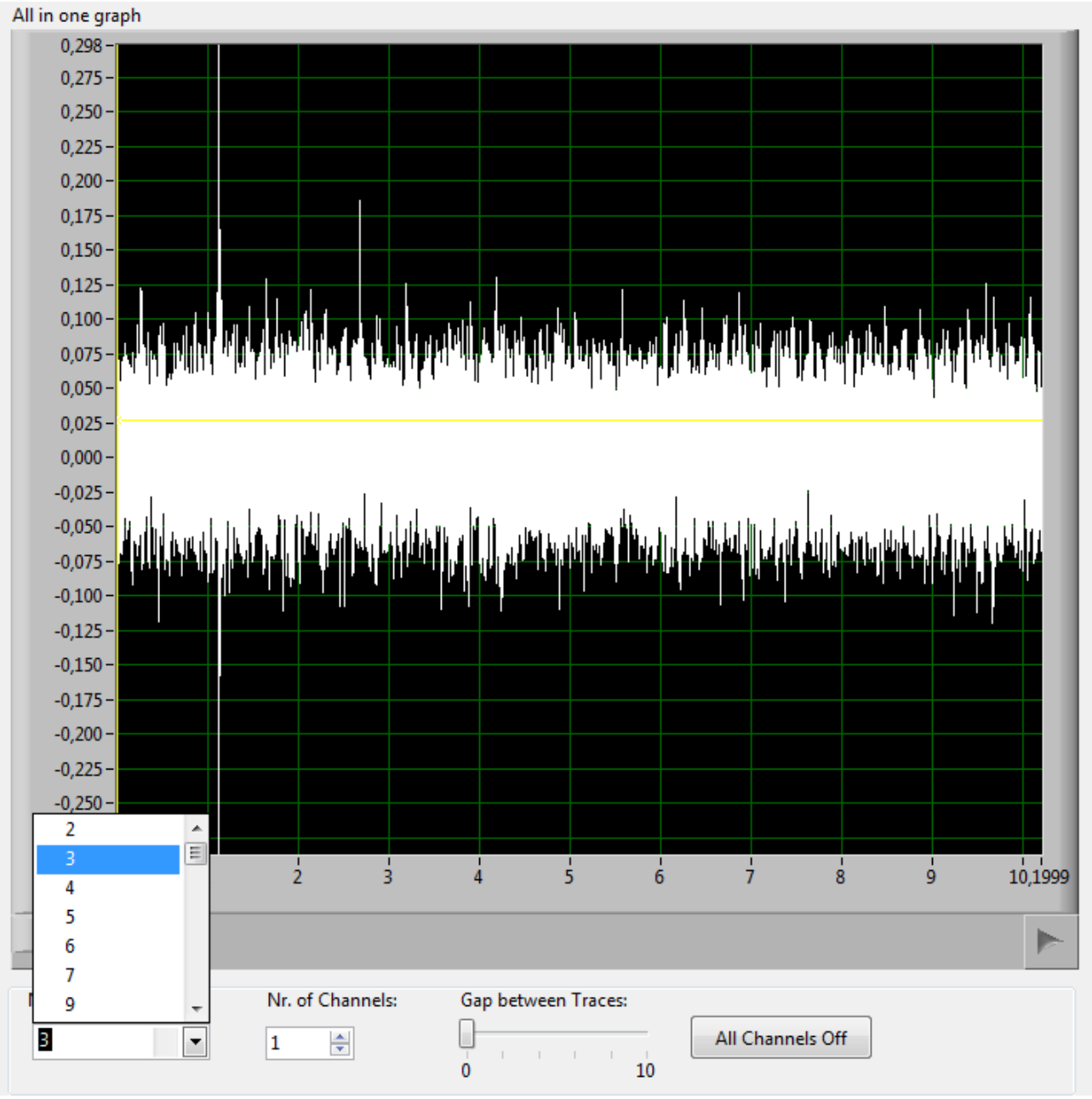



**Figure 55:** Control list while running the program by disabling channel 8.

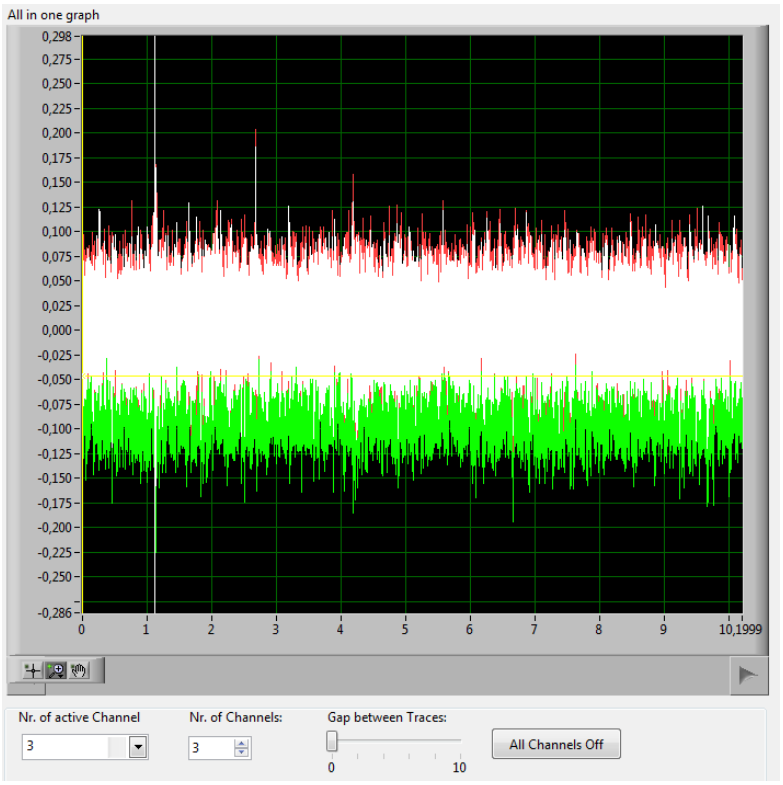



**Figure 56:** Adding channels 4 and 5 to the selected channel 3.

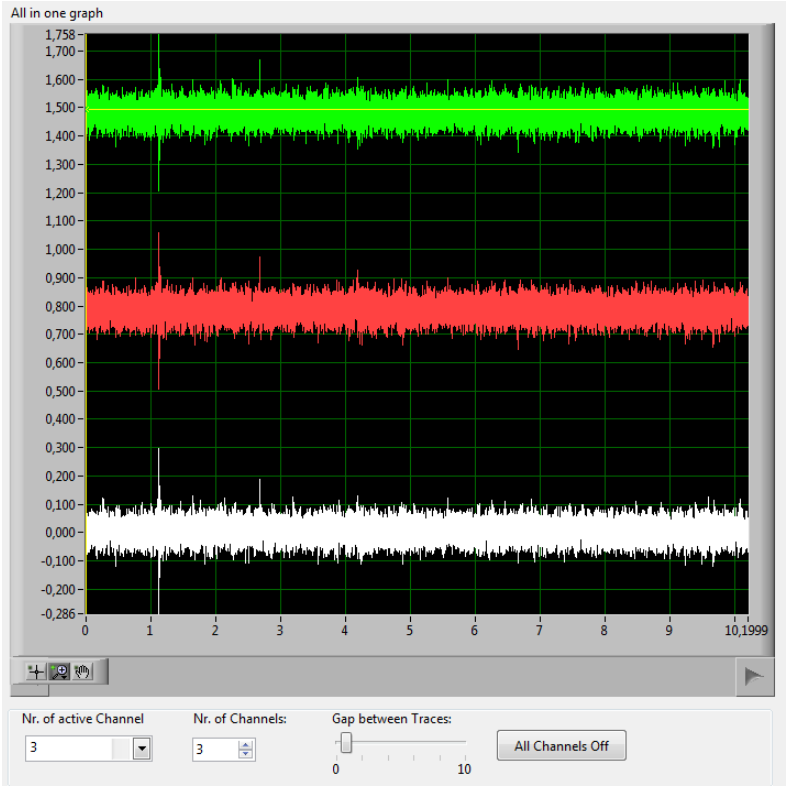



**Figure 57:** Adding a gap between the channels 3, 4 and 5.

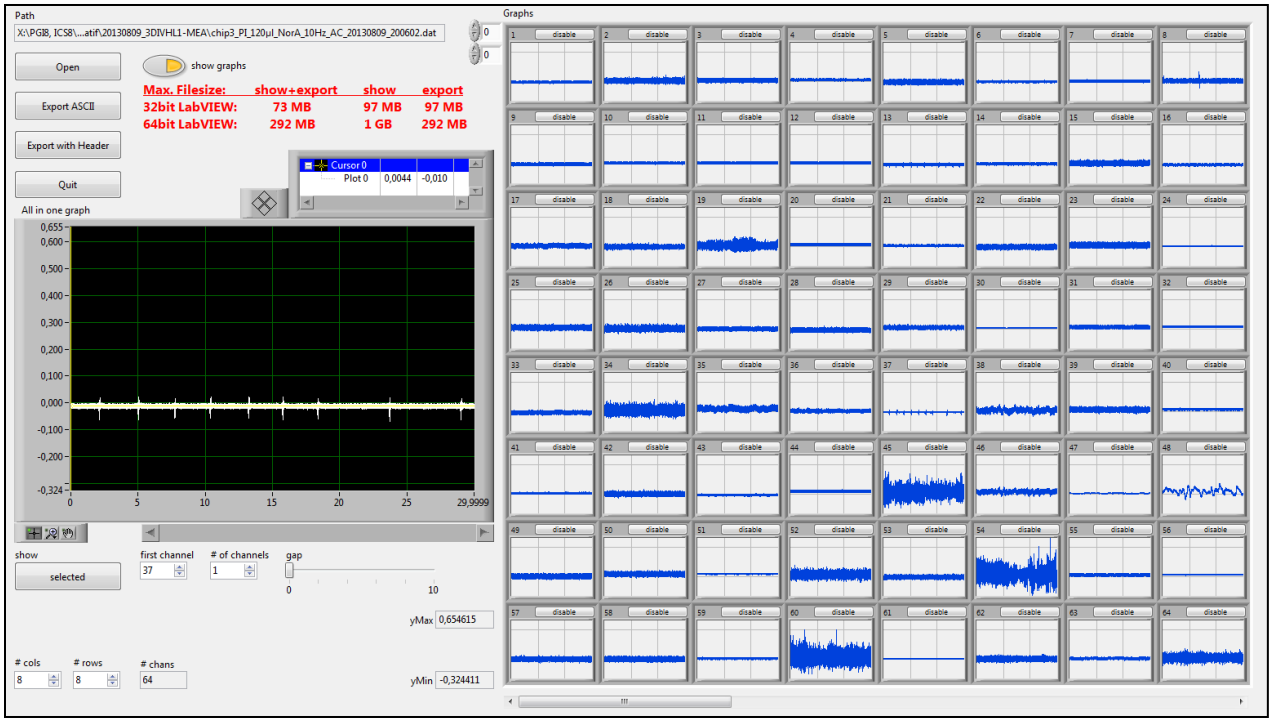



**Figure 58:** Selecting the channel 3 before auto-scaling.

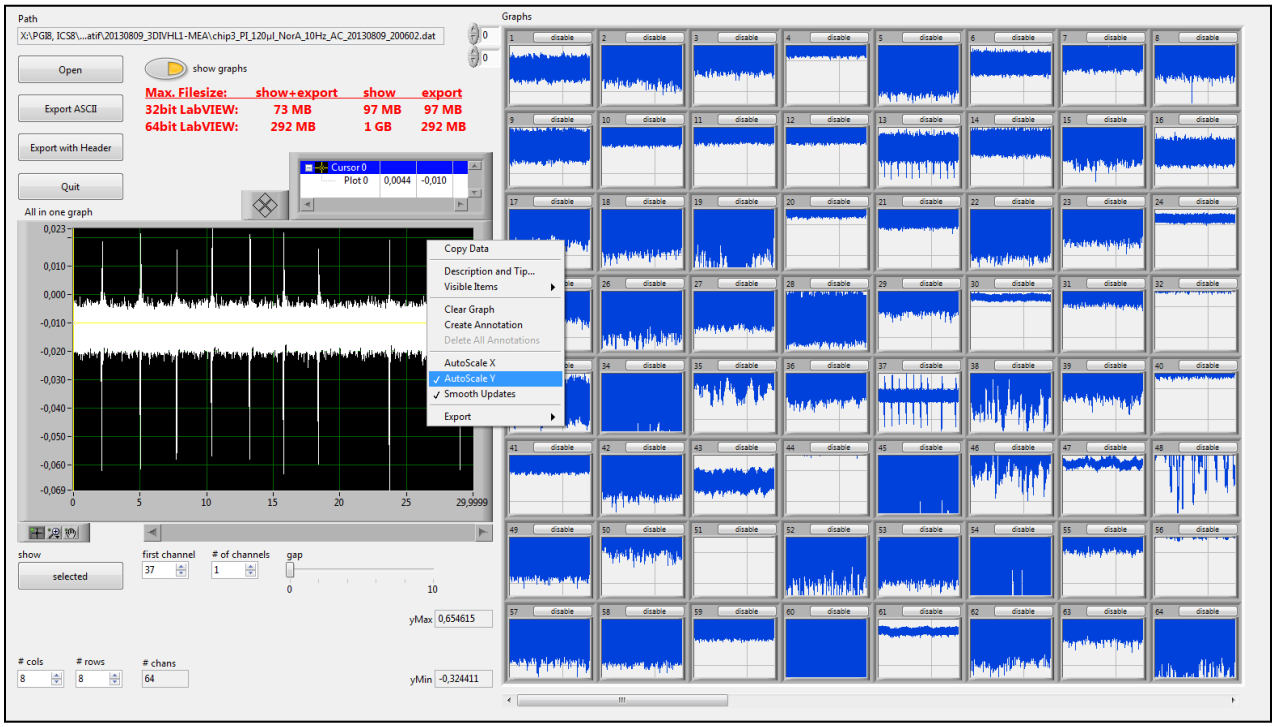



**Figure 59:** Selecting channel 3 after auto-scaling.

## 4.2 Extracellular Recording from Cardiac Cells

An experimental measurement of HL-1 cells was carried out in order to investigate the test of principal of the "Viewer" program by analyzing the voltage traces of the recorded cellular signals. The measurement displayed the signals that were obtained from the electrodes of the MEA chips. In the experiment, the data from a spontaneous activity and the stimulated activity of the cells were compared. The stimulation of the cells was performed using noradrenalin. The resulting data is displayed in the "all in one graph" in the "Viewer" program.

The experiment has been performed with MEA chips (n=5). For example, in chip 3, the signal was displayed during the spontaneous activity (Figure 60). Then, the noradrenalin was pipetted in steps (20 µl, 40 µl, 60 µl, 80 µl, 100 µl, 120 µl). Zooming in channel 37 has occurred in Figure 61 to get a better view of the action potential during spontaneous activity. Figure 62 to Figure 65 show channels 37 and 13 after pipetting 120 µl noradrenalin. This stimulation increased the frequency of the signals.

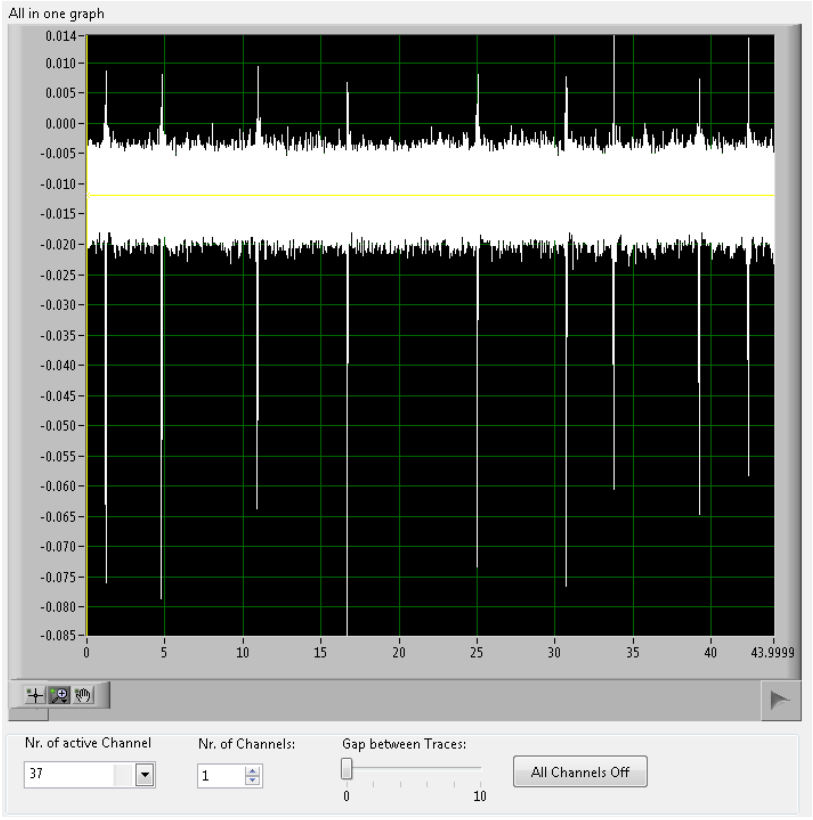



**Figure 60:** Signals of Channel 37 displaying spontaneous activity.

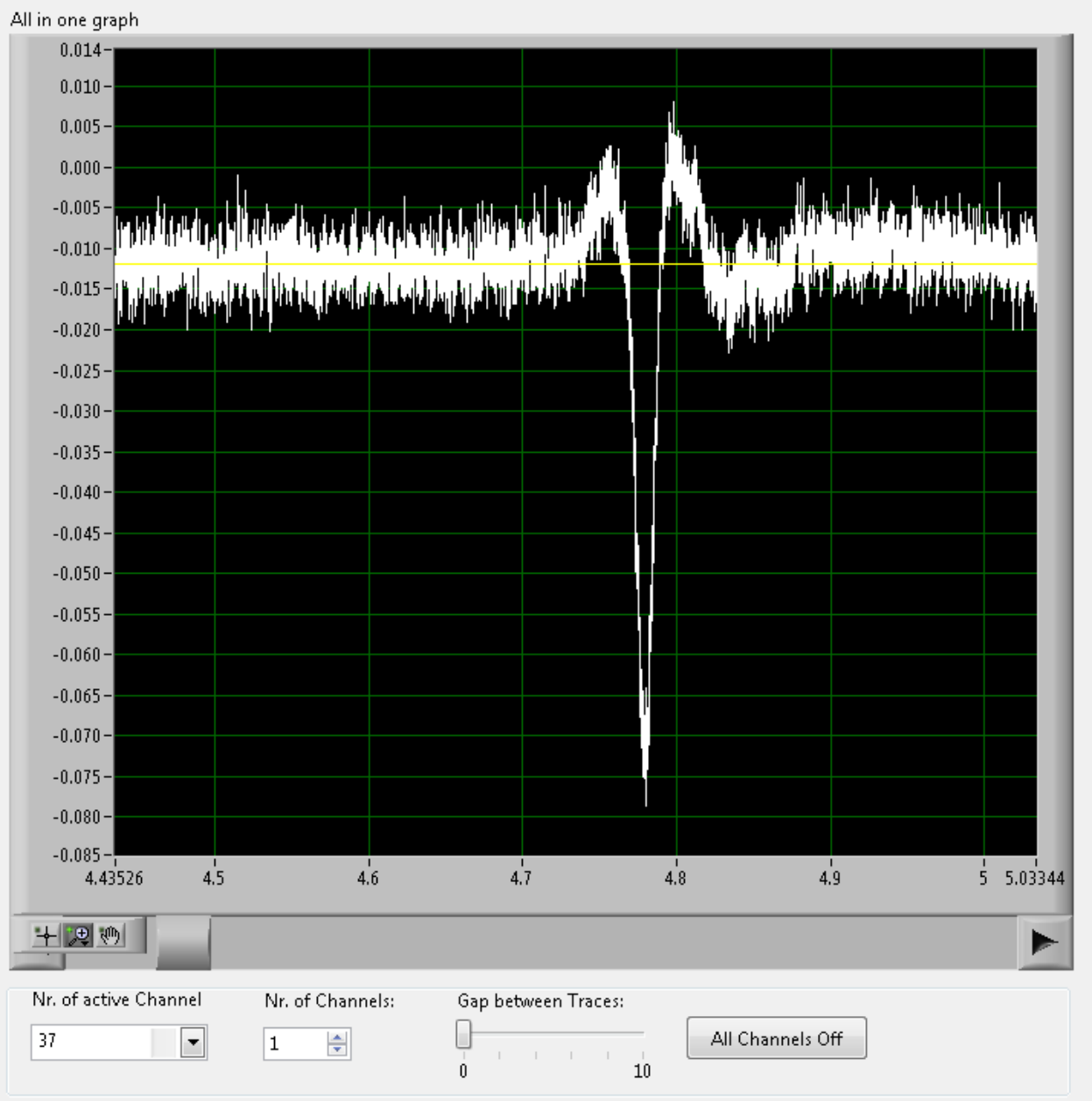



**Figure 61:** Zooming in an action potential signal of channel 37 during spontaneous activity.

With the help of the "all in one graph" in Figure 60 and choosing two sequential peaks, the difference in time between two sequential action potentials during the spontaneous activity can be calculated. The next step was to use the zooming-in function to allow a better view of the two peaks. An example is illustrated in Figure 61, where a single action potential peak has been selected. The same procedure was made to the other sequential peak. The difference in time between two sequential action potentials during

the spontaneous activity was 4.5 seconds and the frequency was 0.22 Hz. The next procedure was to add noradrenalin to the cells, so that the behavior response of the stimulated cells can be tested. After pipetting 120 µl noradrenalin as shown in channel 37 (Figure 62), there is a continuous occurrence of action potential signals in shorter time differences between any two sequential signals, which has a result that the frequency has been increased. The frequency calculated for channel 37 was 0.31 Hz.

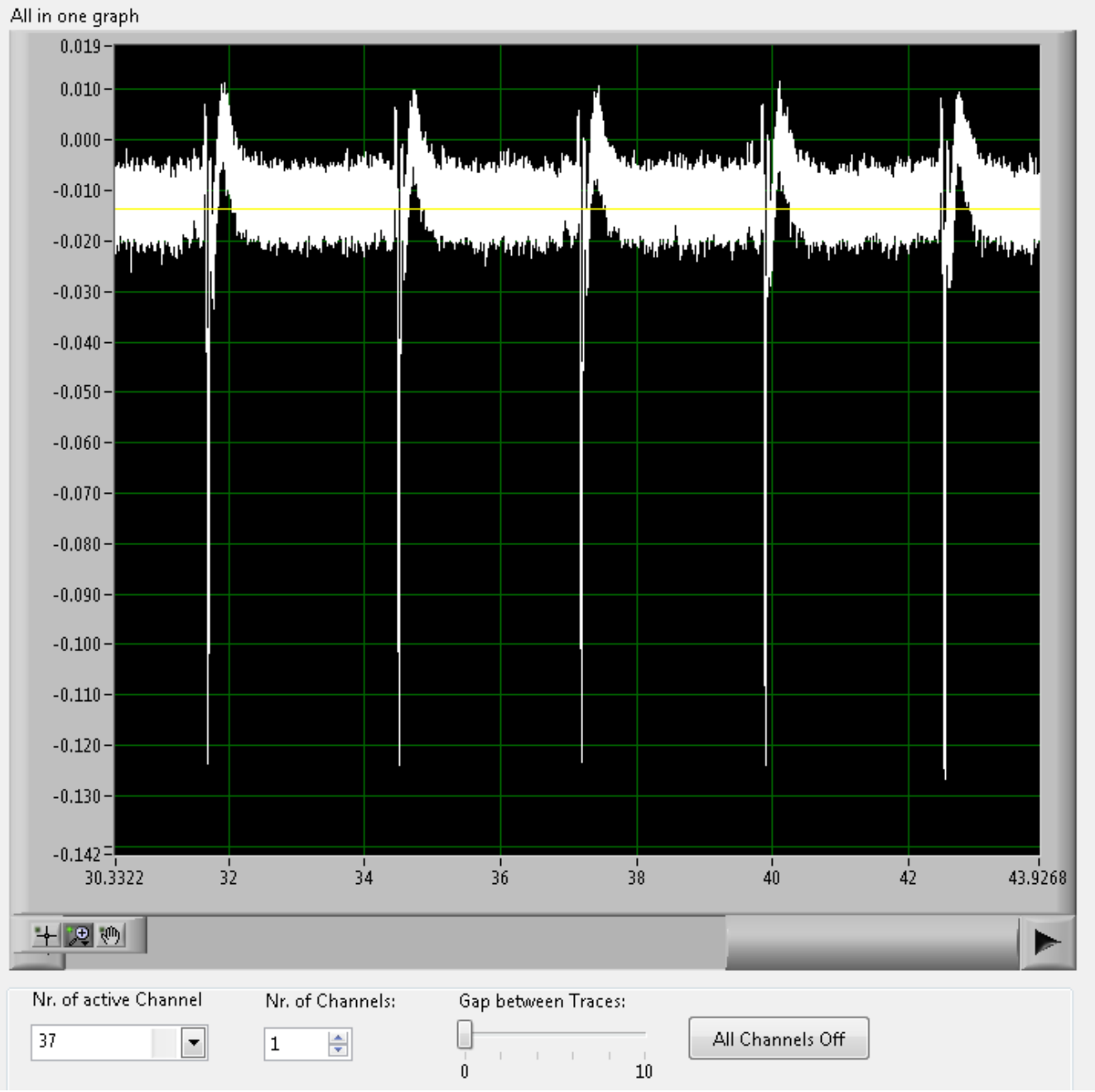



**Figure 62:** Channel 37 after pipetting 120 µl noradrenalin.

The same procedure has been done in channel 13. In Figure 63, it is also shown how fast and tremendous the sequential action potential signals take place. After choosing two sequential peaks, the frequency was 0.29 Hz for channel 13. Figure 64 and Figure 65 are zoomed to clarify the specific action potential time interval chosen in order to compare between channels 13 and 37.

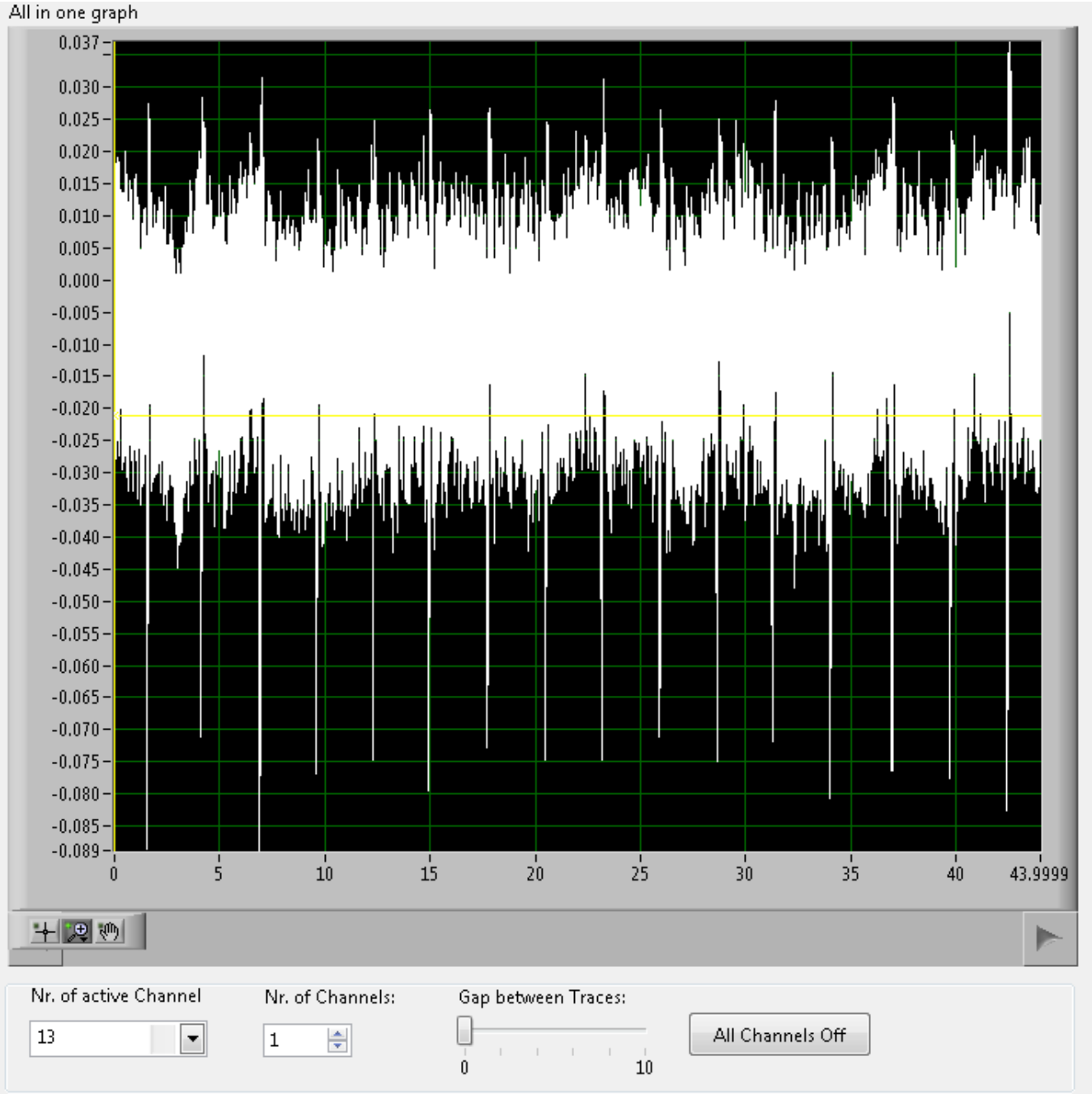



**Figure 63:** Channel 13 after pipetting 120 µl noradrenalin.

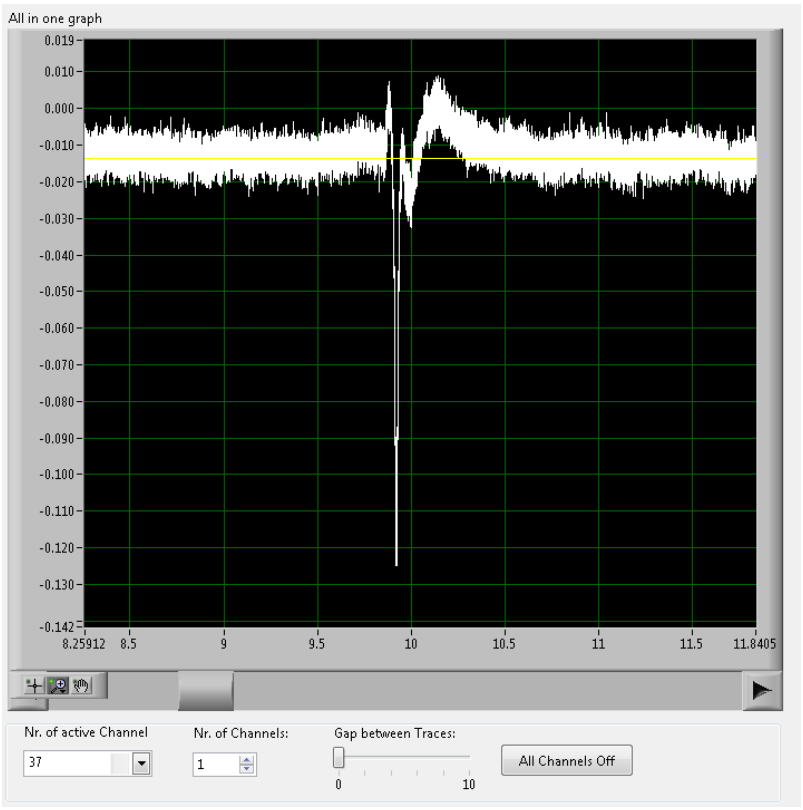



**Figure 64:** Zooming on channel 37 after pipetting 120 µl noradrenalin.

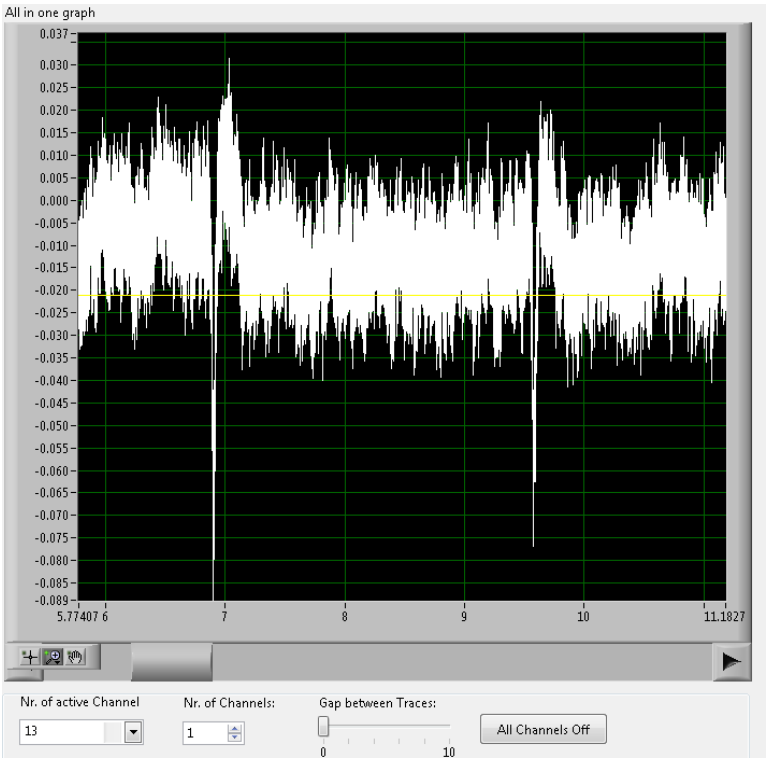



**Figure 65:** Zooming on channel 13 after pipetting 120 µl noradrenalin.

As a conclusion of the comparison of the time intervals between the two peaks in channel 37 and 13 (after pipetting 120 µl noradrenalin) in Figure 64 and 65, the difference in time was 25 ms. Using the difference of the distance (Δx) and the time difference (Δt), the velocity of an action potential can be calculated with the equation $v = \frac{\Delta x}{\Delta t}$. The MEA schematic is shown in Figure 66. The dark points are the electrodes. The distance between each horizontal or vertical electrode is 200 µm. The distance between electrode 13 and 37 is 600 µm.

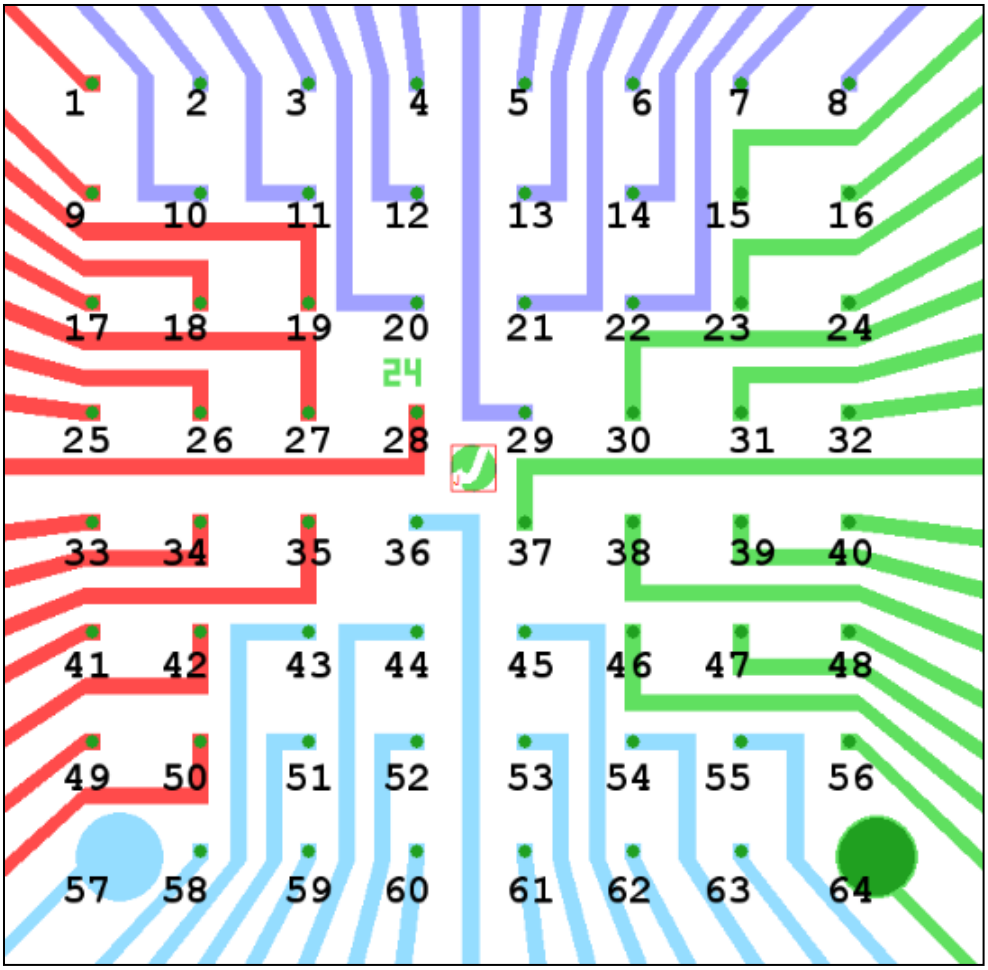



**Figure 66:** Schematic view of a MEA chip.

After calculating the difference in time between the two peaks and the corresponding distance between channel 13 and 37, the propagation velocity was 24 $\frac{\text{mm}}{\text{s}}$. This is within the same range that was obtained for the measurement of the velocity propagation of HL-1 cells by Eick and Machinek [16,21].

## 4.3 PyAnalyzer

The "PyAnalyzer" is a program that was developed using the programming language Python. This programming language is used for a variety of scientific purposes and has many advantages: it is an open source (free of charge), high performance language that is also an easily readable, dynamic-typed language.

Active development, especially for science-related tasks, uses the scientific-related modules of Python "Numpy" and "Scipy" for the spike sorting and spike-train analysis projects [21].

Due to elementary limitations in the memory consumption that occur while analyzing typical data sets of electrophysiological measurements[1], it seemed feasible to rewrite the viewer program (**chapter 4.1**). One of the main objectives of the program was to start the development of an efficient platform for electrophysiology analysis captured by the in-house developed multipurpose "BioMAS" amplifier system.

Before the cellular signals were analyzed in detail, the development of the "PyAnalyzer" was initiated with efficient plotting functions, which is equivalent to the main purpose of the LabView "Viewer" program, and additionally offers rudimentary noise-analysis functions, such as the evaluation of the Root-Mean-Square (RMS) noise, Peak-to-Peak noise, and spectral analysis functions (FFT).

As mentioned in **chapter 2.3.2**, the Python source code was developed with a specialized Integrated Development Environment (IDE) "Spyder" (Figure 13) offering a MATLAB®-like editor. There are many libraries available that focus on a scientific use, for example, "Numpy" (for efficient numerical data handling), with highly efficient libraries featuring large sets of array objects [21].

Calculating the "Peak to peak" value in "Python" can be achieved by importing "Numpy" and calling the "numpy.ptp(*variable*)" function, where *variable* is an input array value. For the "Fourier-transform" of the recorded data, libraries such as "scipy.fft()," which returns the discrete "Fourier-transform" of real or complex sequences from the SciPy-module, were used (Figure 67).

[1] typically in the range of hundreds of MB

To demonstrate the usage of the "PyAnalyzer", a large set of random data was generated. The RMS value for the signal (Figure 68) as well as the upper limit (3 x sigma) and the lower limit (-3 x sigma) were calculated and plotted with "Matplotlib." This specia l module was written for Python to plot graphs. The complete code, written in Python, is in the Appendix B.

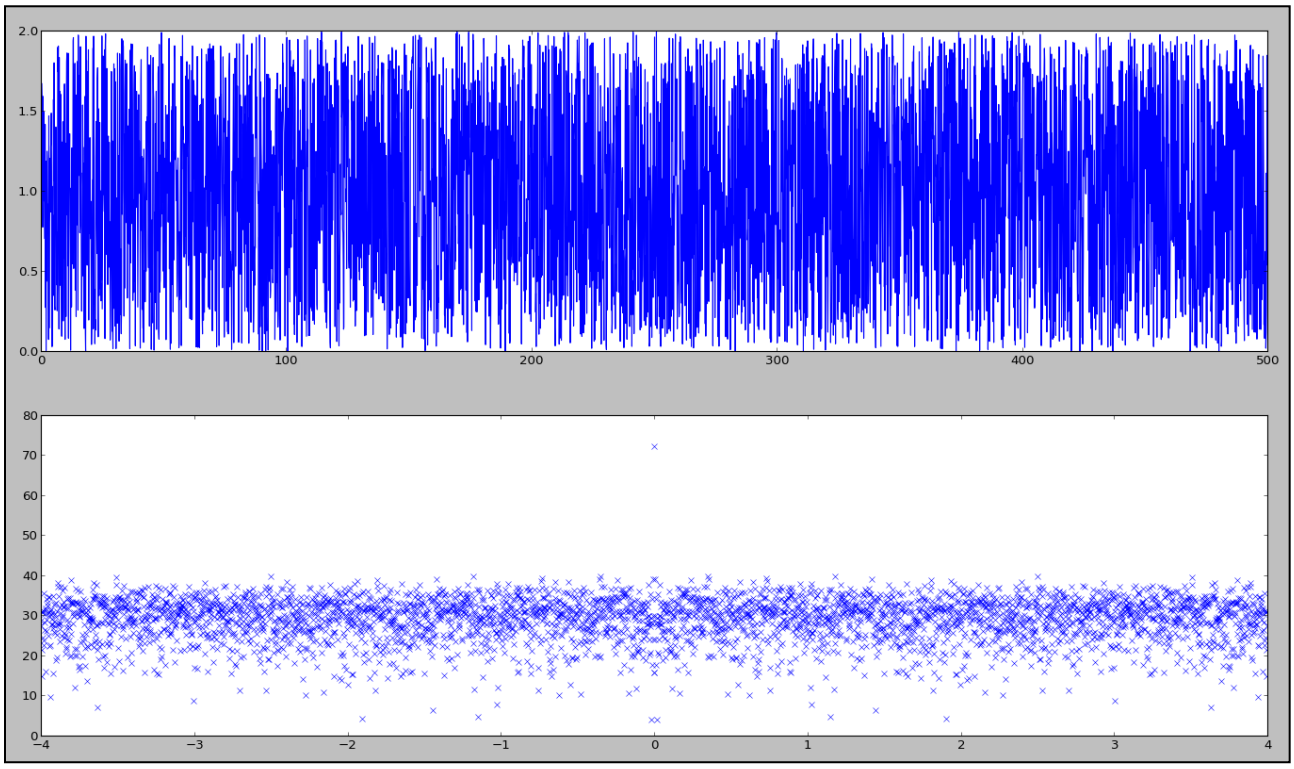



**Figure 67:** Applying the function "scipy.fft()" to the random noise signal.

A further calculation of the Power-Spectral-Density (PSD) was carried out and is shown in Figure 69. This was achieved with the function "plt.psd()," which was imported from the module "matplotlib.pyplot".

Python QT (Figure 70) was used to build up the graphical user interface (GUI) for a comfortable handling of data sets. With the Python QT module one can create several applications, such as buttons, charts, lists, and checkboxes. It is similar to the front panel in the LabVIEW program. After creating the necessary objects, the connection between the objects takes place to send the required information. In cooperation with the Python code, functions can be implemented inside the corresponding objects.

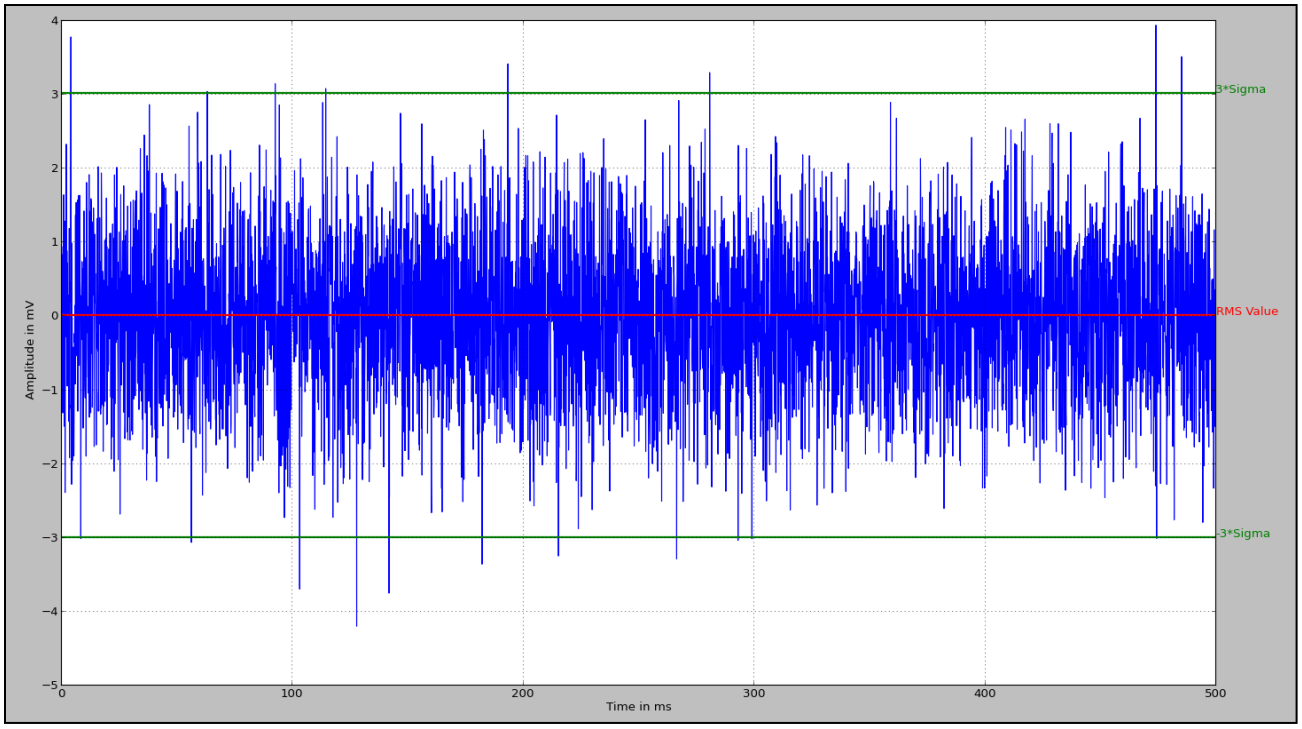



**Figure 68:** Plot of the amplitude, sigma range and RMS value of the noise signal.

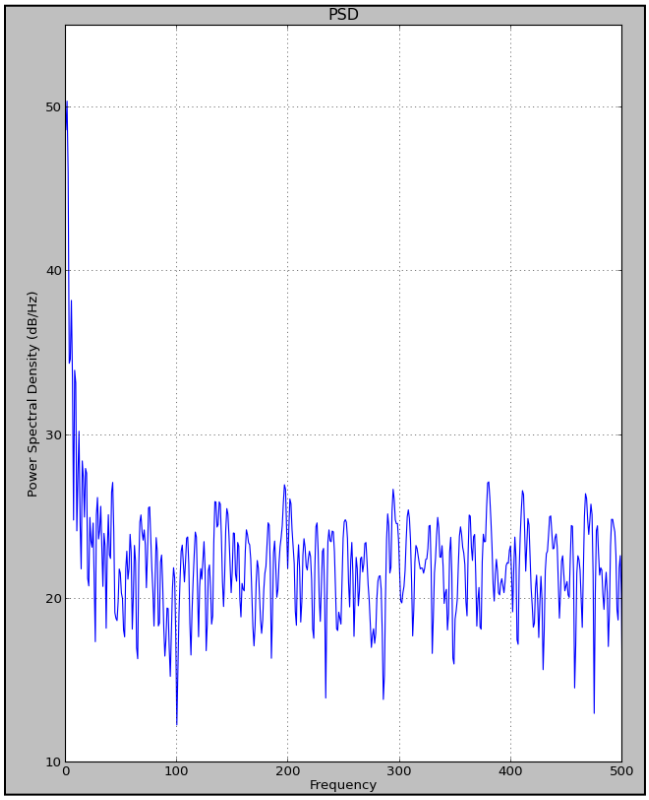



**Figure 69:** Plot of the Power-Spectral-Density (PSD) of a random noise function.

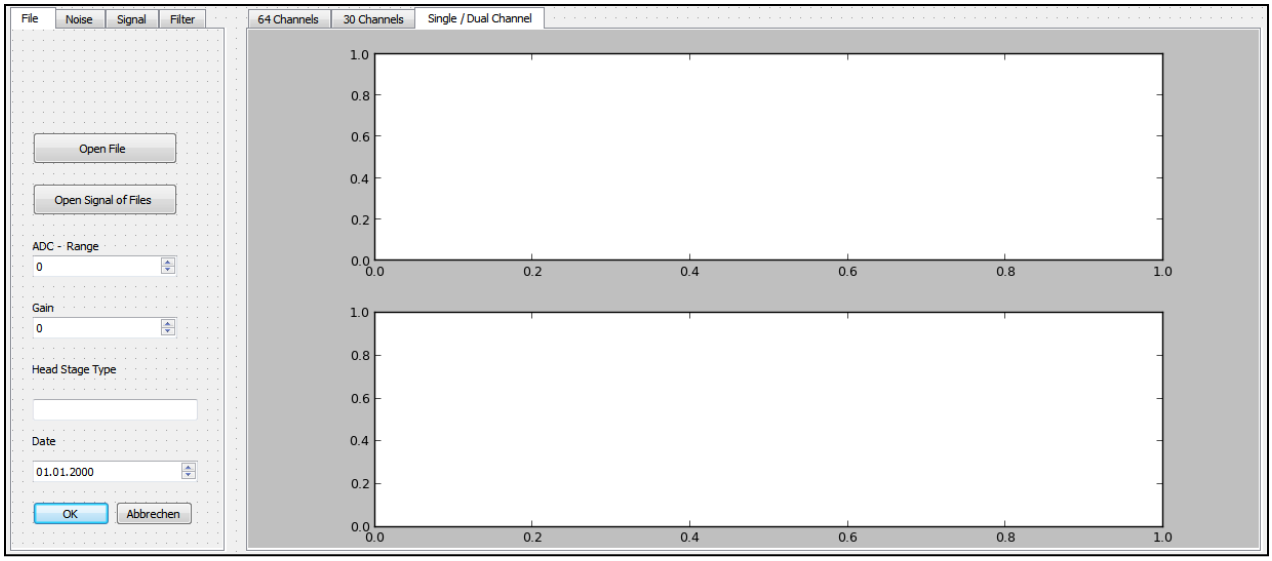



**Figure 70:** Python QT framework for the graphical interface of the PyAnalyzer.

# 5. Conclusions and Outlook

## 5.1 Conclusion

During this thesis, several optimizations to increase the functionality of this tool were performed:

- Displaying the missing channels.
- Ordering the channels in the right arrangement.
- Enabling the manual disabling of the individual channels.
- Illustrating the maximum/minimum values of the individual channels.
- Enabling/disabling the graphs.
- Optimizing the efficiency of data handling.
- Optimization of the exporting function.

In conclusion, the "Viewer" program is able to import the acquired data and plot each channel in individual graphs. The option of plotting all the channels in one graph was enabled. The orientation of the channels has been corrected by using the active channels.

The user can choose to display any desired channel individually in the "all in one graph." In order to make the display and the differences between the channels clearer, the "Gap between Traces" function was introduced.

Especially challenging was how to save memory as the graphs in the program were using large memory spaces. A comparison between the single data files notifies the allowed maximum size of the input file that can be imported.

After displaying the data, an export function was installed to allow the export of data for further analysis. The export function makes it possible for the user to export the files with the corresponding headers so that it is clear which channels are active.

In **chapter 4.1.3** it was shown in the 32-bit LabVIEW program that the maximum file to be loaded and exported was 97 MB. For loading and exporting in the 64-bit, the maximum input file was 292 MB. The maximum file to be loaded and plotted in the 32-bit LabVIEW program is 97 MB. For the 64-bit enviroment, the program was able to plot and load files up to 1 GB.

In the 32-bit LabVIEW program, the maximum file to be loaded, exported and plotted was 73 MB. For loading, exporting and plotting in the 64-bit LabVIEW routine the maximum input file was 292 MB. This was written (in red) as a notice for the user in the block diagram of the main program (Figure 71).

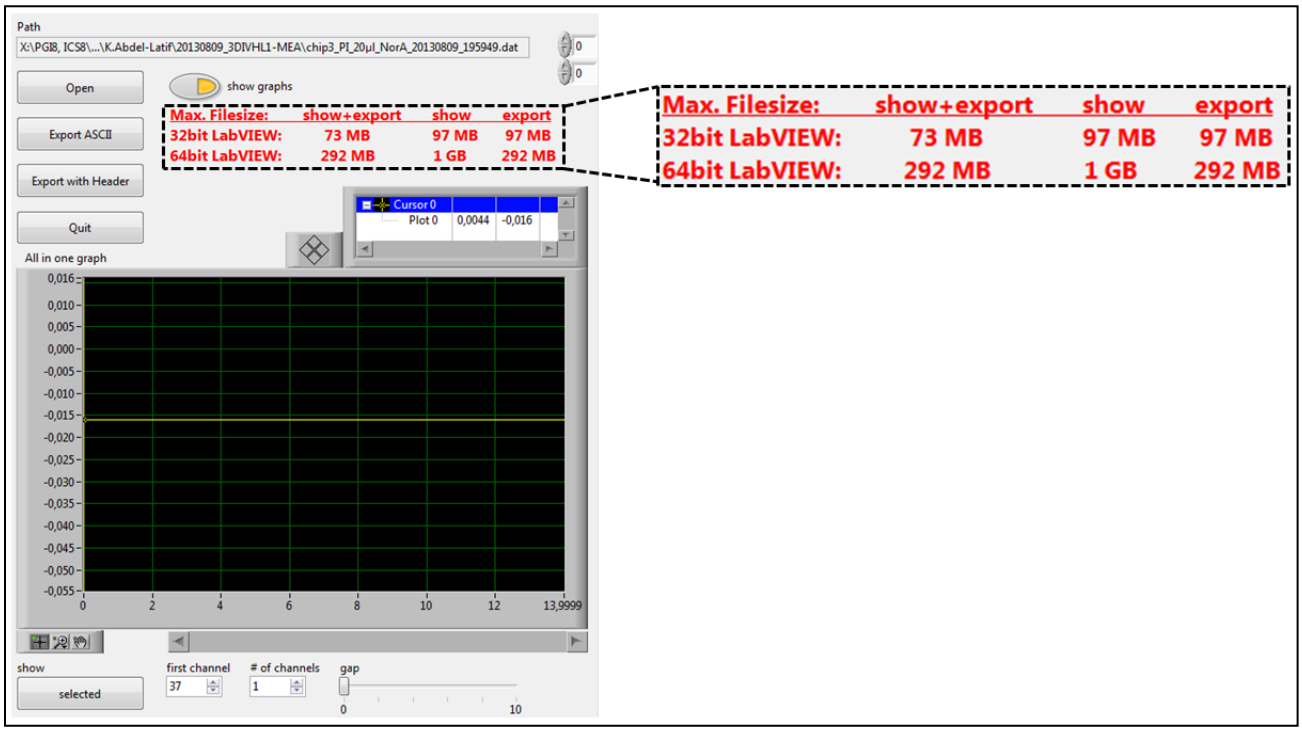



**Figure 71:** The notification (in red) for the user in the main program.

## 5.2 Outlook

For the "Viewer" program, it would be very useful to create an additional list to select more than one channel and plot them together in the "all in one graph" chart so that the user has the choice to select two or more channels arbitrarily.

The function "nr. of channel" was able to add channels which lie after the selected channel. For example, when a channel is selected, then only the sequential channels can be added. Nevertheless, in the future it would be useful to select, for example, channel 3 and 10 directly together.

The function "nr. of channel" could be exchanged with another list of "nr. of active channel 2". The user could then choose the desired two channels to be compared. The gap between channels could be modified automatically so

that as soon as the user selects the two different channels, the gap function should display the two channels with a relative way as the automatic scaling.

In the future, “Python” will play a distinct role in importing, analysing and efficiently plotting data. There are many examples showing that “Python” is applied in electrophysiology. Python was used for spike sorting and spike train analysis in different projects [21]. There have been tests, carried out on the efficiency of Python for high-performance computing. “There are hardly any efficiency loss by doing the message passing in Python” [22].

# Bibliography

[1] G. G. Matthews: *Neurobiology* : *Molecules, Cells, and Systems*, Blackwell Science, Malden, MA, pp. 10-25 (1998)

[2] H. James: *The Edwin Smith Surgical Papyrus: Hieroglyphic Transliteration, Translation and Commentary, Kessinger Publishing, Band 1*, vol. 9, pp. 430-435 (2006)

[3] E. Solomon, L. Berg, and D. W. Martin: *Biology*, Brooks Cole,Thomson Learning, pp. 103-130 (6. Edition, 2002)

[4] N. Bursac, M. Papadaki, R. J. Cohen, F. J. Schoen, S. R. Eisenberg, R. Carrier, G. Vunjak-Novakovic, and L. E. Freed: Cardiac muscle tissue engineering: toward an in vitro model for electrophysiological studies, *The American Journal of Physiology*, vol. 277, no. 2, pp. 433-444 (1999)

[5] J. Barnes, *Essential Biological Psychology*. Hardcover: 332 pages, Sage Publications Ltd London., pp. 1-40 (2013)

[6] G. Cook: Membrane structure and function, *Annual Review of Plant Physiology*, pp. 125-141 (1971)

[7] E. Kandel, J. Schwartz, and T. Jessell: *Principles of Neural Science,* McGraw-Hill, pp. 13-22 (4. Edition, 2000)

[8] S. Eick: Extracellular Stimulation of Individual Electrogenic Cells with Micro-Scaled Electrodes, Ph.D.Thesis , RWTH Aachen (2010)

[9] P. Bergveld: ISFET, theory and practice, *IEEE Sensor Conference Toronto*, no. October, pp. 1-26 (2003)

[10] M. J. Schöning: Vorlesungsskript - Biosensorik, FH-Aachen, Standort Jülich (WS 2010/2011)

[11] P. H. Borcherds: Python: a language for computational physics, *Computer Physics Communications*, vol. 177, no. 1-2, pp. 199-201 (2007)

[12] A. Mechtley and R. Trowbridge: *Maya Python for Games and Film*, CRC Press; Elsevier (1. Edition, 2012)

[13] J. R. Johansson, P. D. Nation, and F. Nori, QuTiP: An open-source Python framework for the dynamics of open quantum systems, *Computer Physics Communications*, vol. 183, no. 8, pp. 1760-1772 (2012)

[14] L. Rodríguez-Liñares, A. J. Méndez, M. J. Lado, D. N. Olivieri, X. A. Vila, and I. Gómez-Conde: An open source tool for heart rate variability spectral analysis, *Computer Methods and Programs in Biomedicine*, vol. 103, no. 1, pp. 39-50 (2011)

[15] W. C. Claycomb, N. A. Lanson, B. S. Stallworth, D. B. Egeland, J. B. Delcarpio, A. Bahinski, and N. J. Izzo: HL-1 cells: a cardiac muscle cell line that contracts and retains phenotypic characteristics of the adult cardiomyocyte, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 95, no. 6, pp. 2979-2984 (1998)

[16] T. Anmann, R. Guzun, N. Beraud, S. Pelloux, A. V Kuznetsov, L. Kogerman, T. Kaambre, P. Sikk, K. Paju, N. Peet, E. Seppet, C. Ojeda, Y. Tourneur, and V. Saks: Different kinetics of the regulation of respiration in permeabilized cardiomyocytes and in HL-1 cardiac cells. Importance of cell structure/organization for respiration regulation , *Biochimica et Biophysica Acta*, vol. 1757, no. 12, pp. 1597-606 (2006)

[17] M. Jansen: Silizium Nanoribbon Feld-Effekt-Transistoren zur Kopplung an elektroaktive Zellen, Ph.D.Thesis, RWTH Aachen (2013)

[18] D. Brüggemann, B. Wolfrum, V. Maybeck, Y. Mourzina, M. Jansen, and A. Offenhäusser: Nanostructured gold microelectrodes for extracellular recording from electrogenic cells, *Nanotechnology*, vol. 22, no. 26, pp. 265104 (2011)

[19] W. S. N. Shim, S. Jiang, P. Wong, J. Tan, Y. L. Chua, Y. S. Tan, Y. K. Sin, C. H. Lim, T. Chua, M. Teh, T. C. Liu, and E. Sim: Ex vivo differentiation of human adult bone marrow stem cells into cardiomyocyte-like cells, *Biochemical and Biophysical Research Communications*, vol. 324, no. 2, pp. 481-488 (2004)

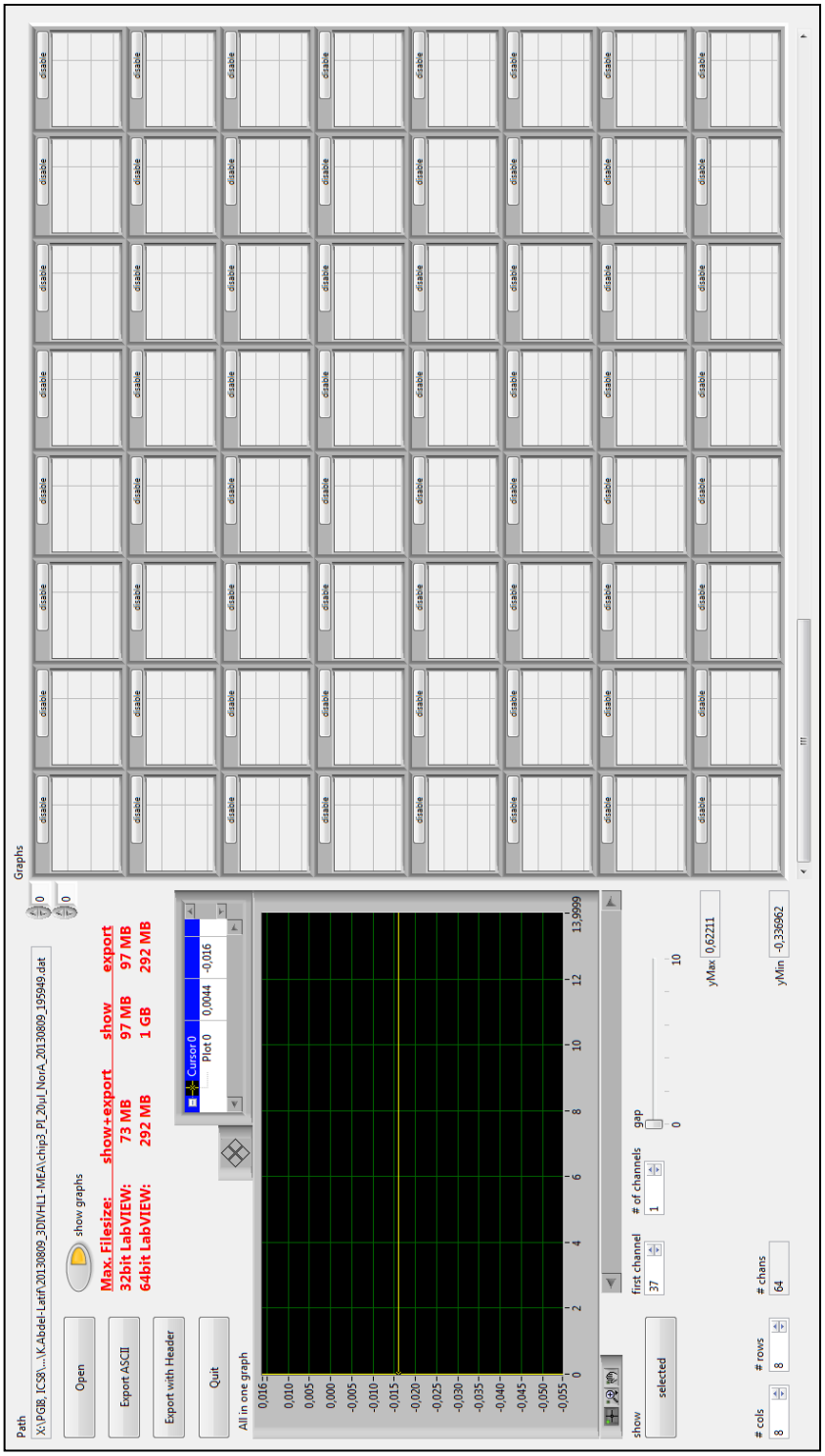
[20] R. Machinek: *Stimulation and Recording of Electrical Cell Signals using Electronic Devices*, B.Sc.Thesis, RWTH Aachen (2010)

[21] M. Spacek, T. Blanche, and N. Swindale: *Python for large-scale electrophysiology* , *Frontiers in Neuroinformatics*, vol. 2, no. January, pp. 1-6 (2008)

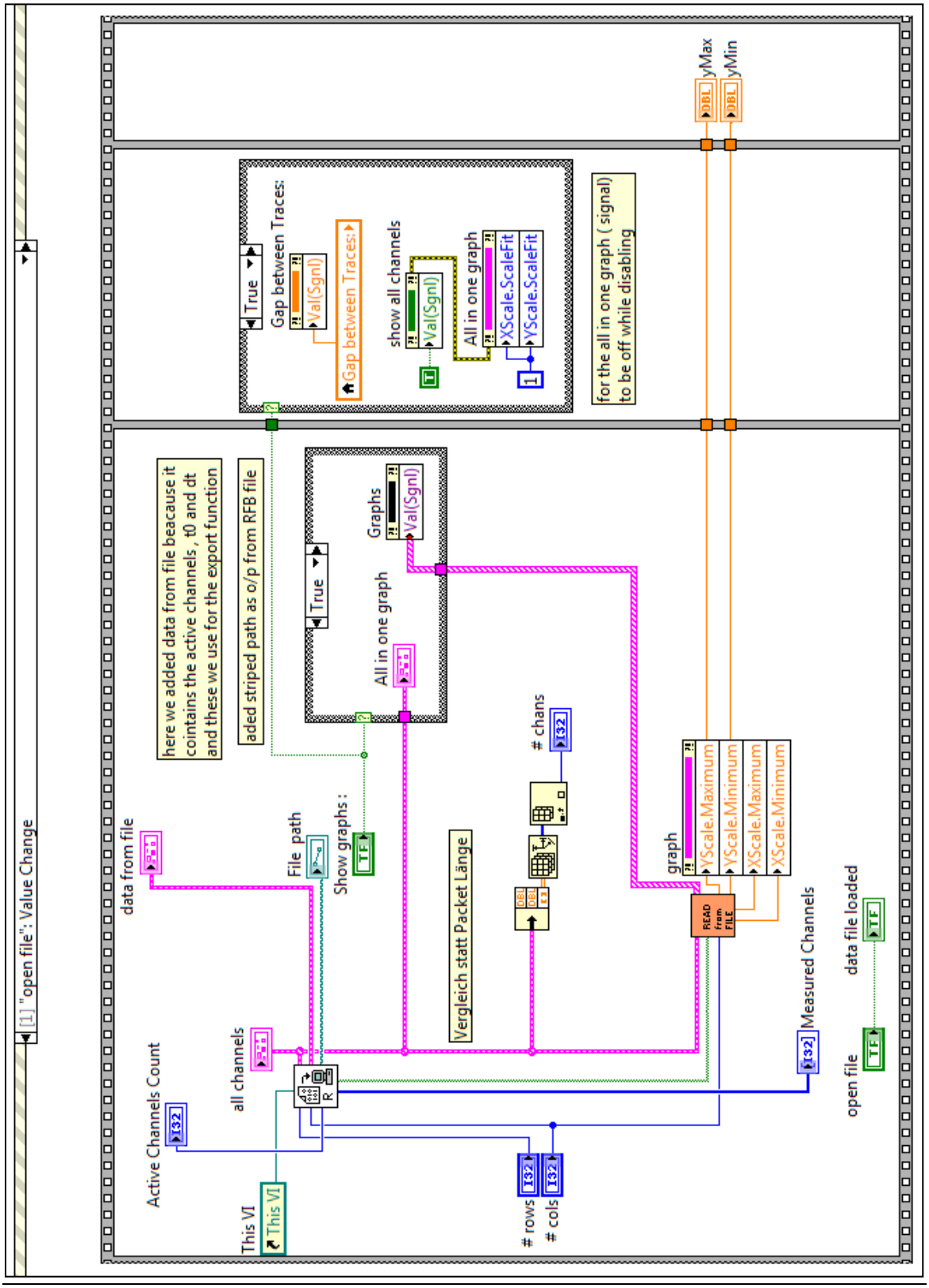
[22] H. G. Bock, E. Kostina, H. X. Phu, and R. Rannacher: *Modeling, Simulation and Optimization of Complex Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 220-231 (2008)

# Appendix A: The Viewer

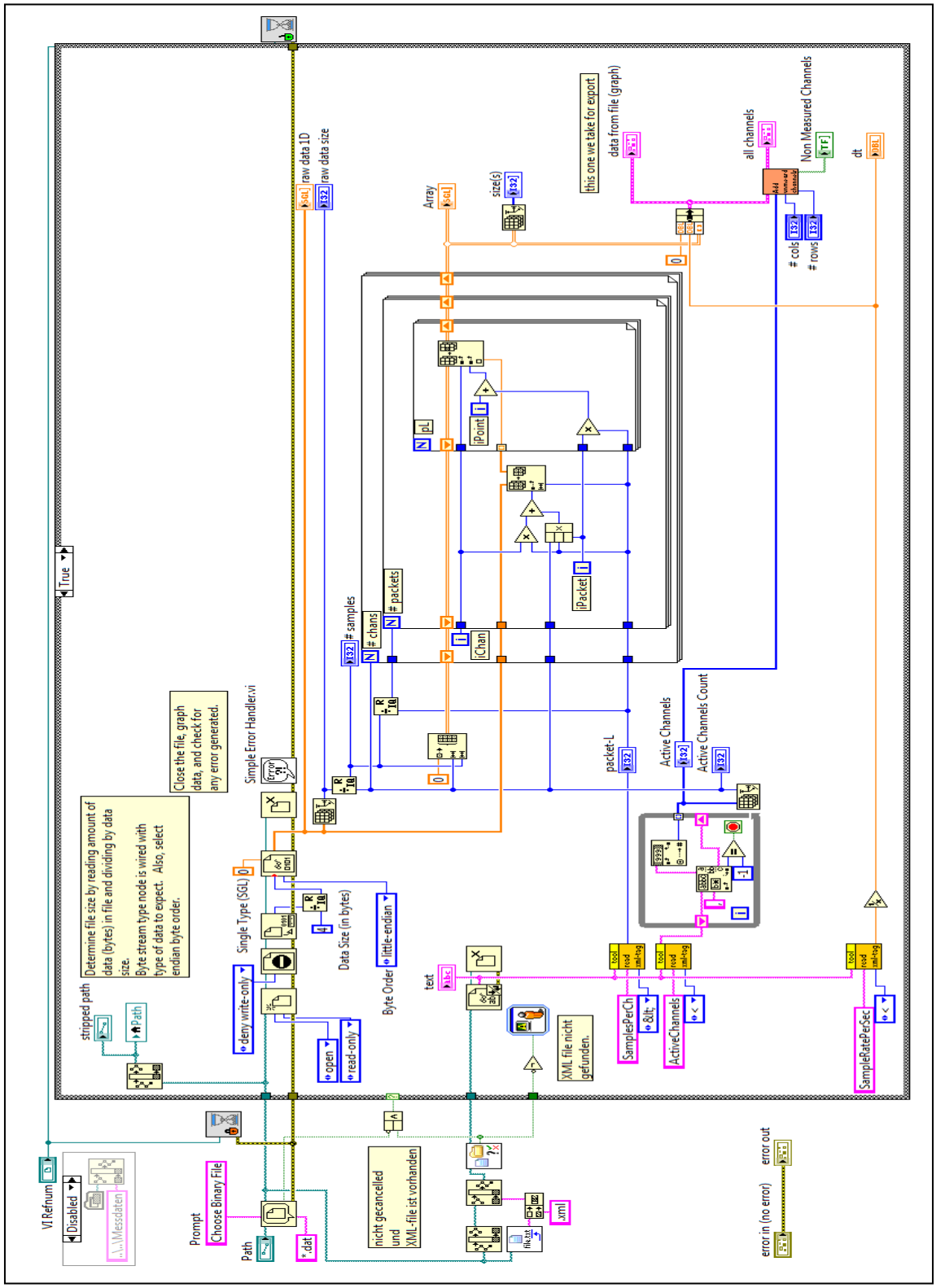
The front panel of the main program.



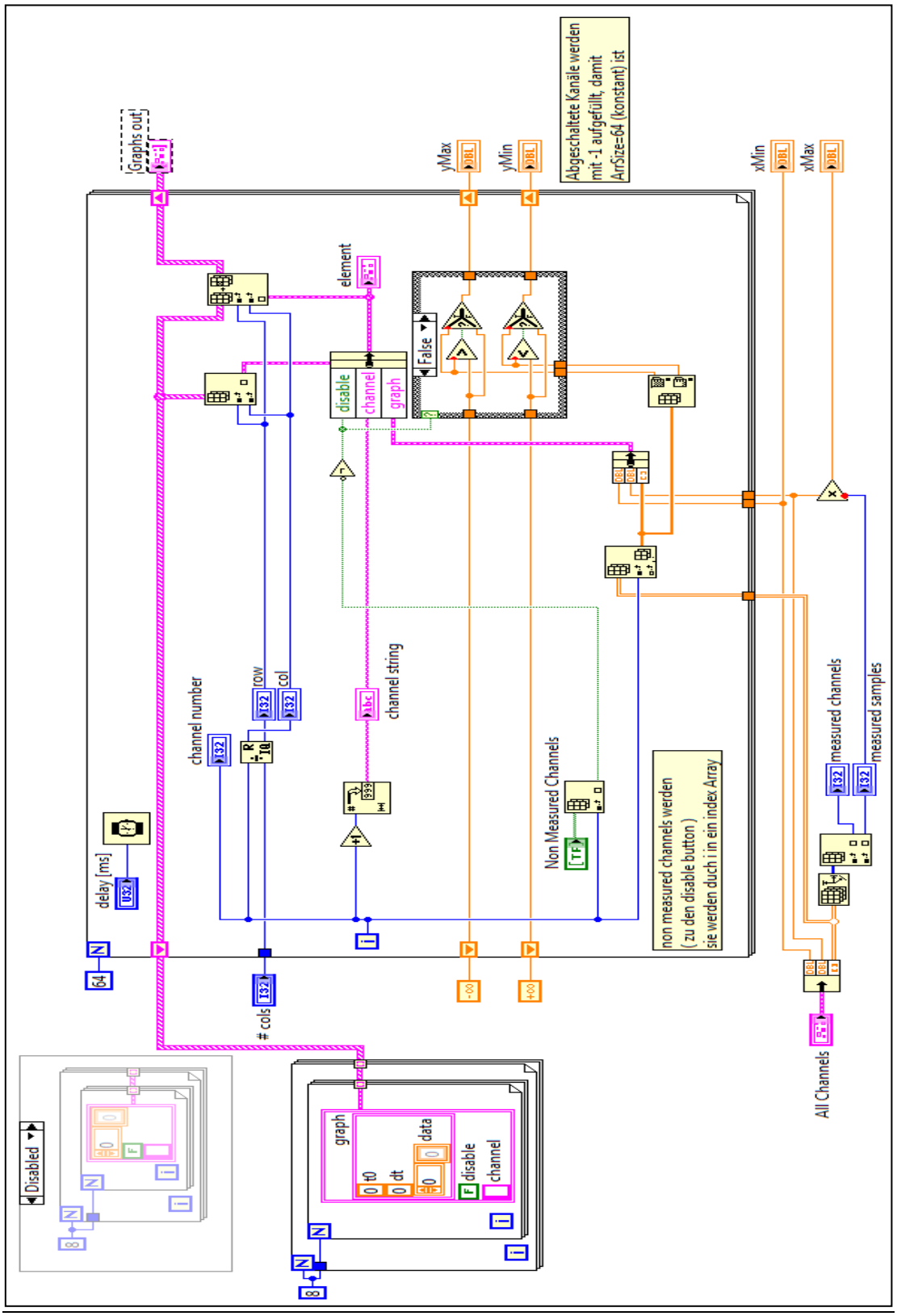
The "open file event":



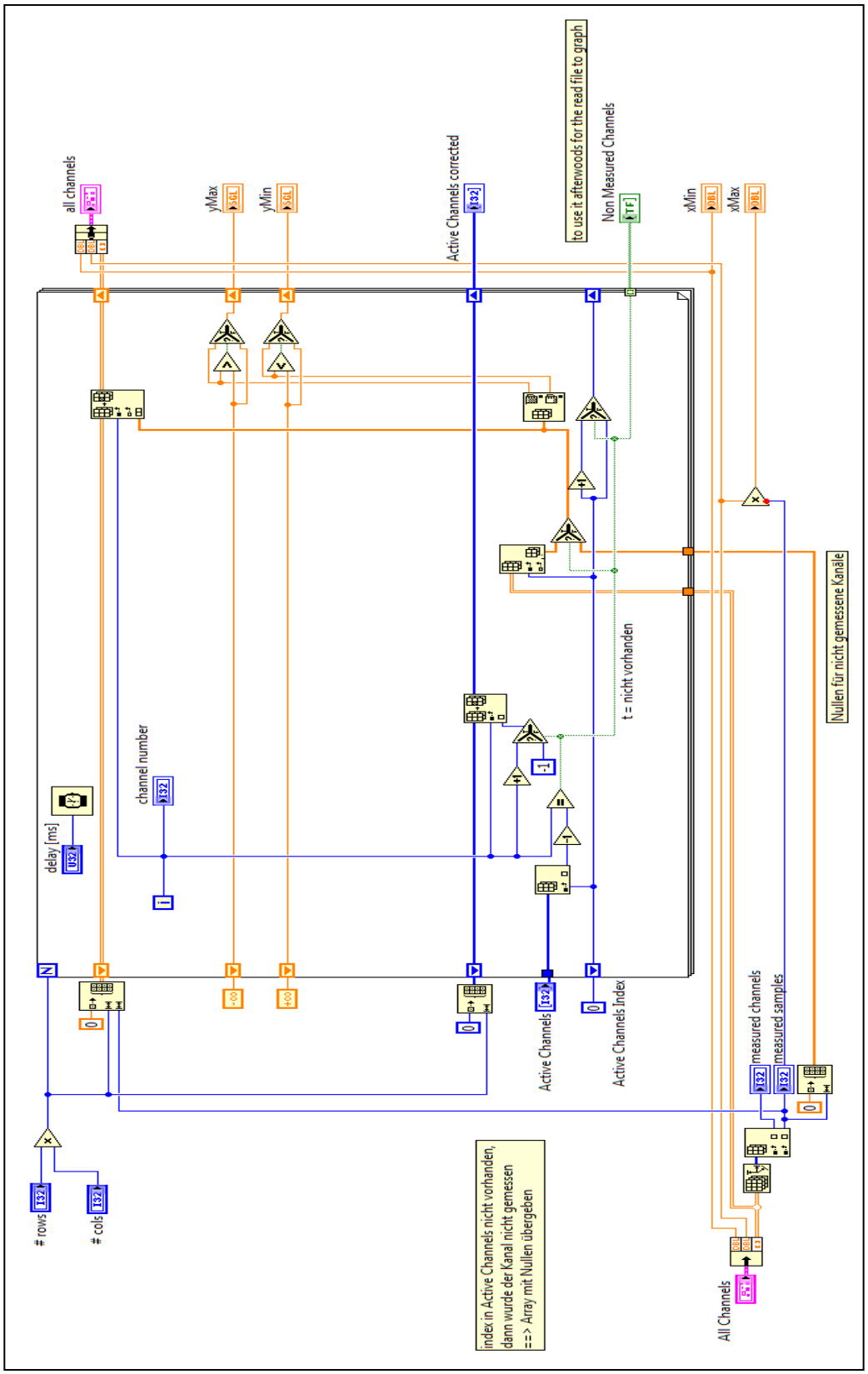
"Read from binary SubVI":



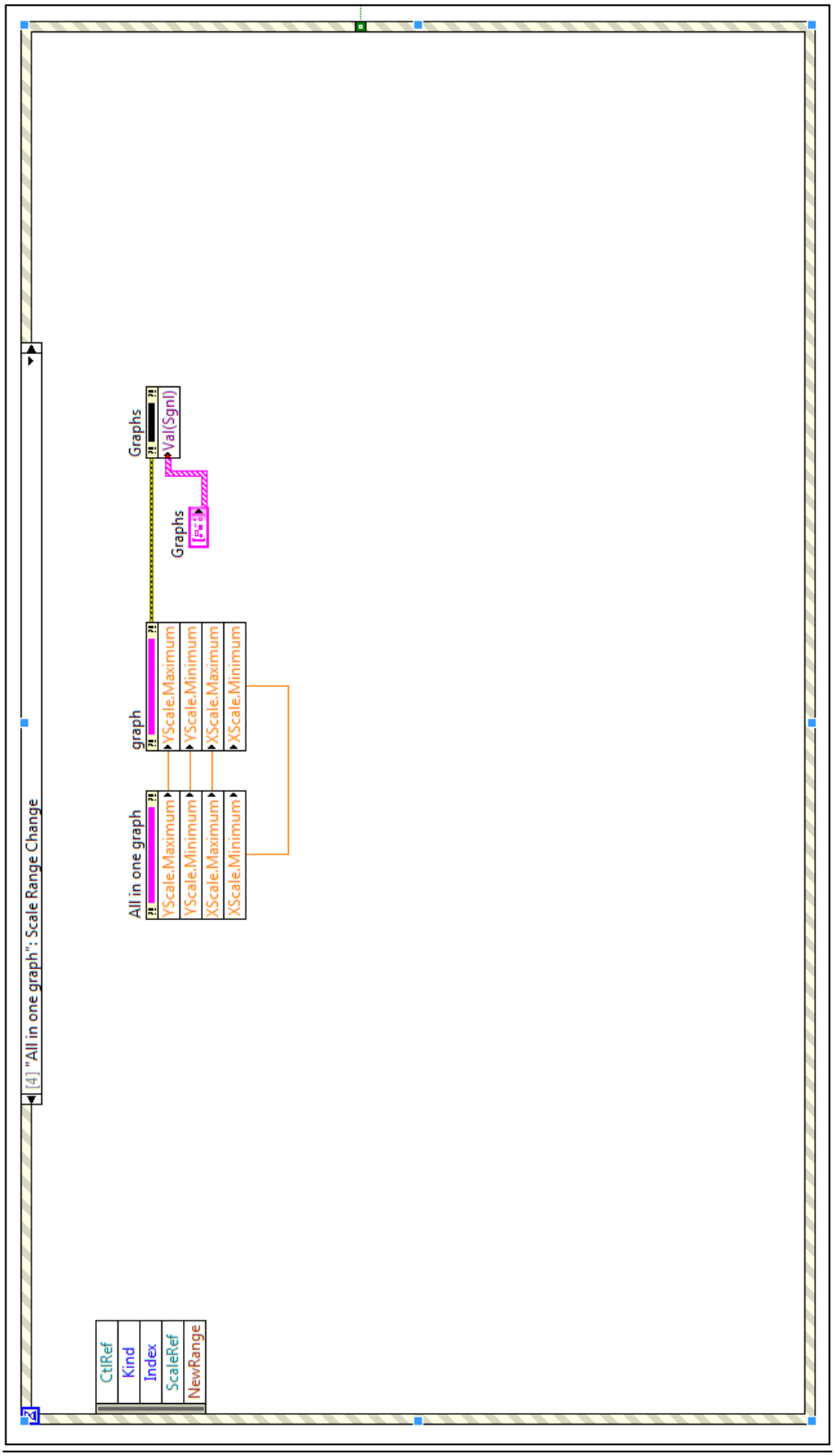
"Read file to graph SubVI":



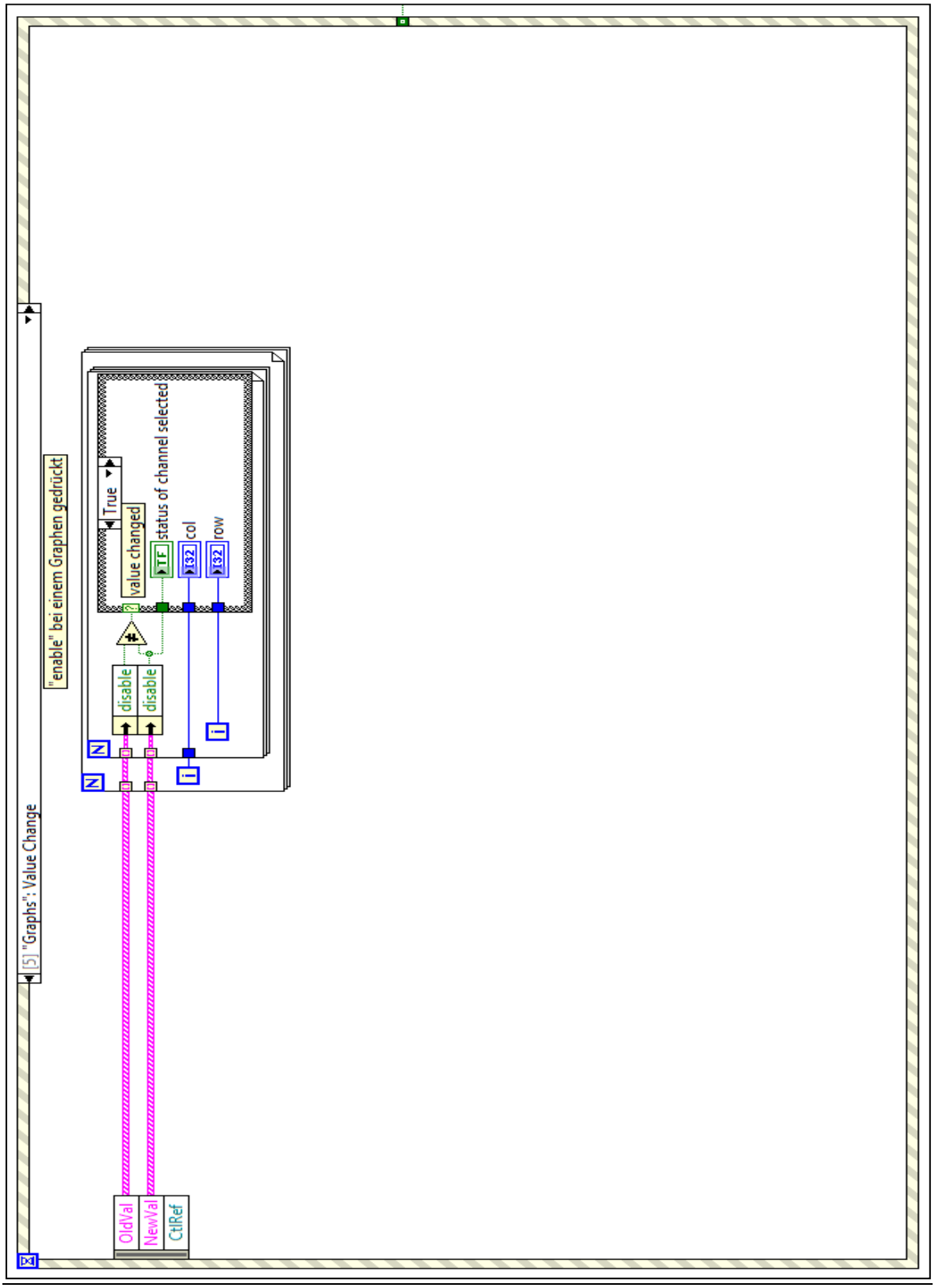
"Add disabled channels SubVI":



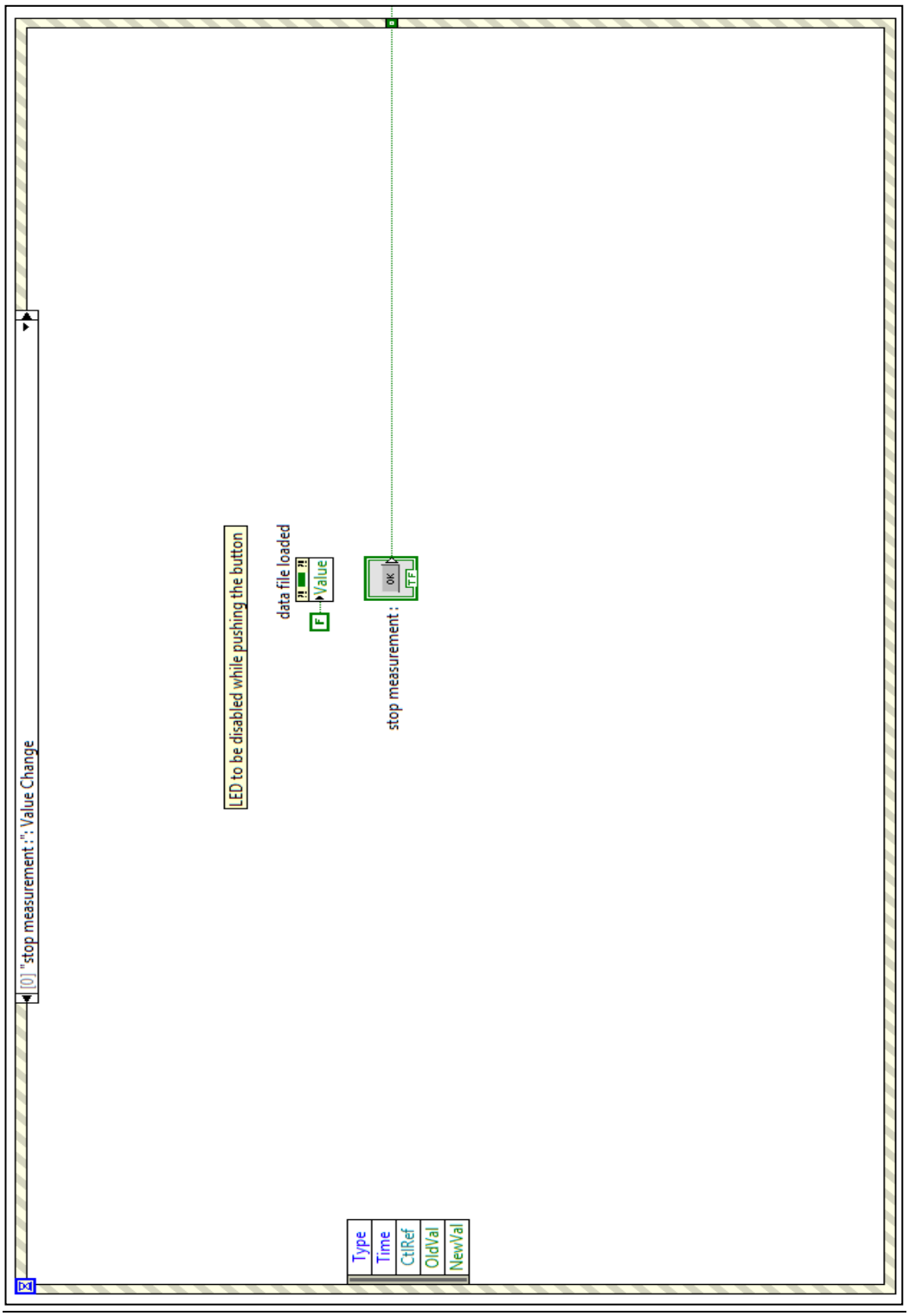
"Scale range change event":



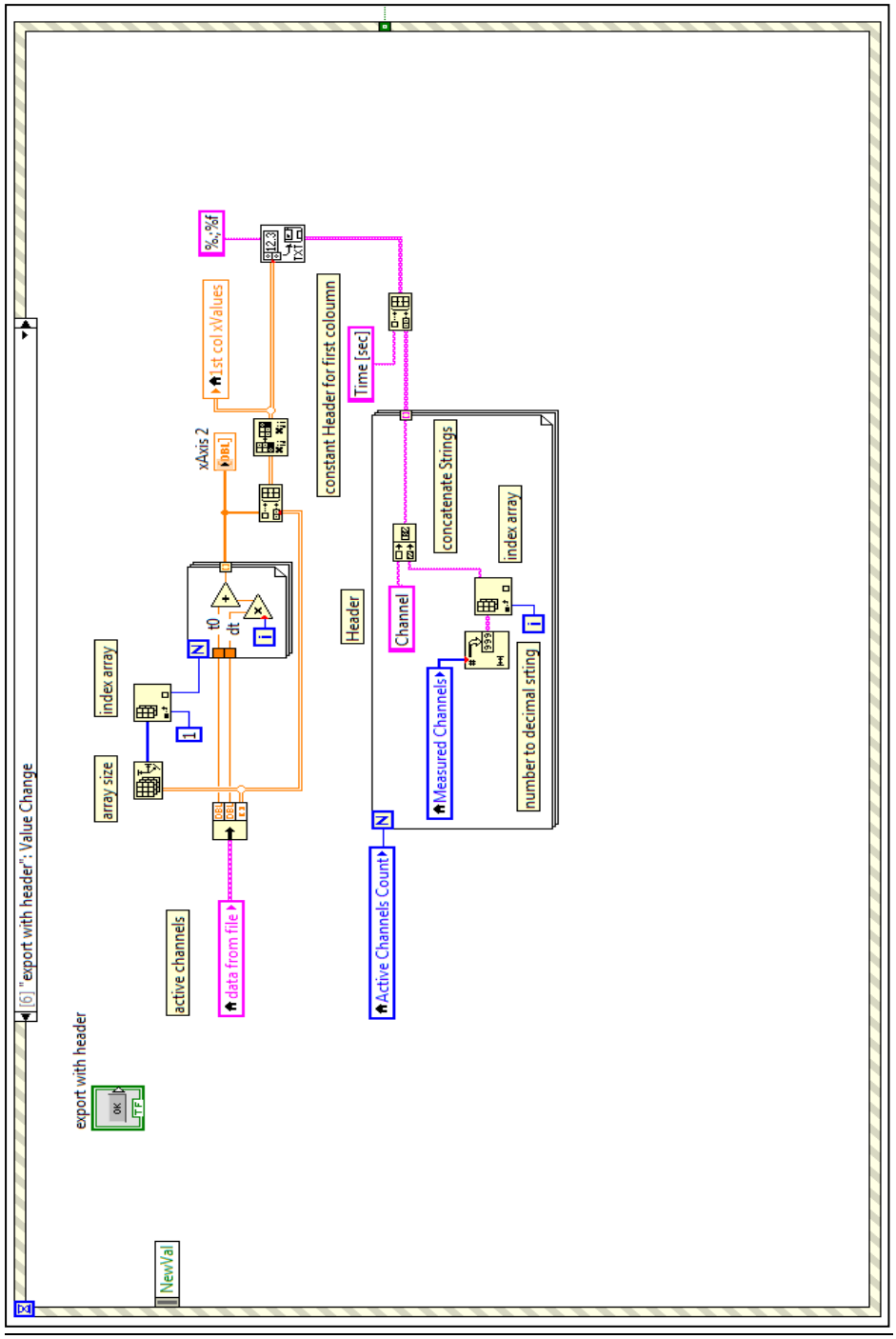
"Graph value change event":



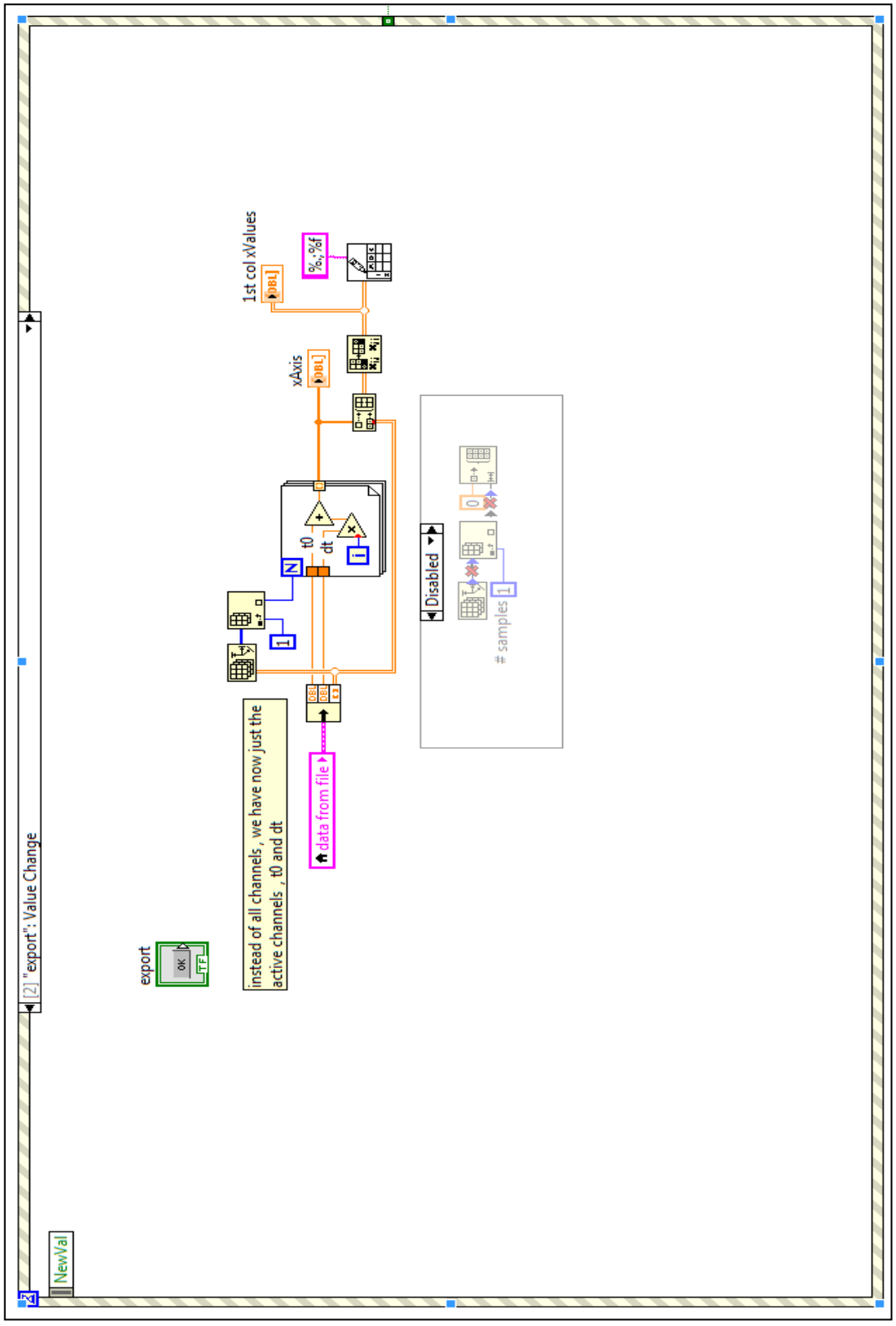
"Stop measurement event":



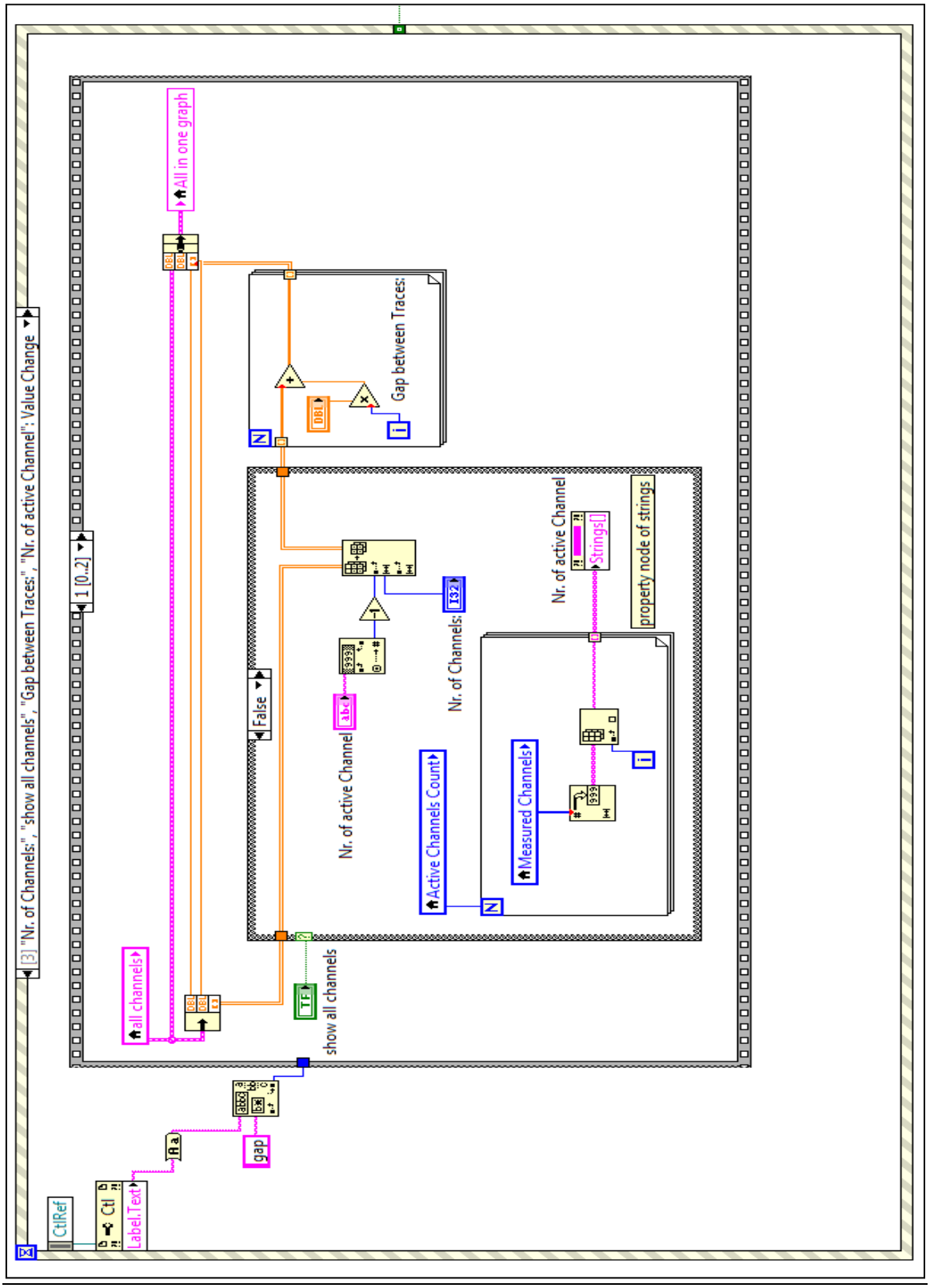
"Export with header event":



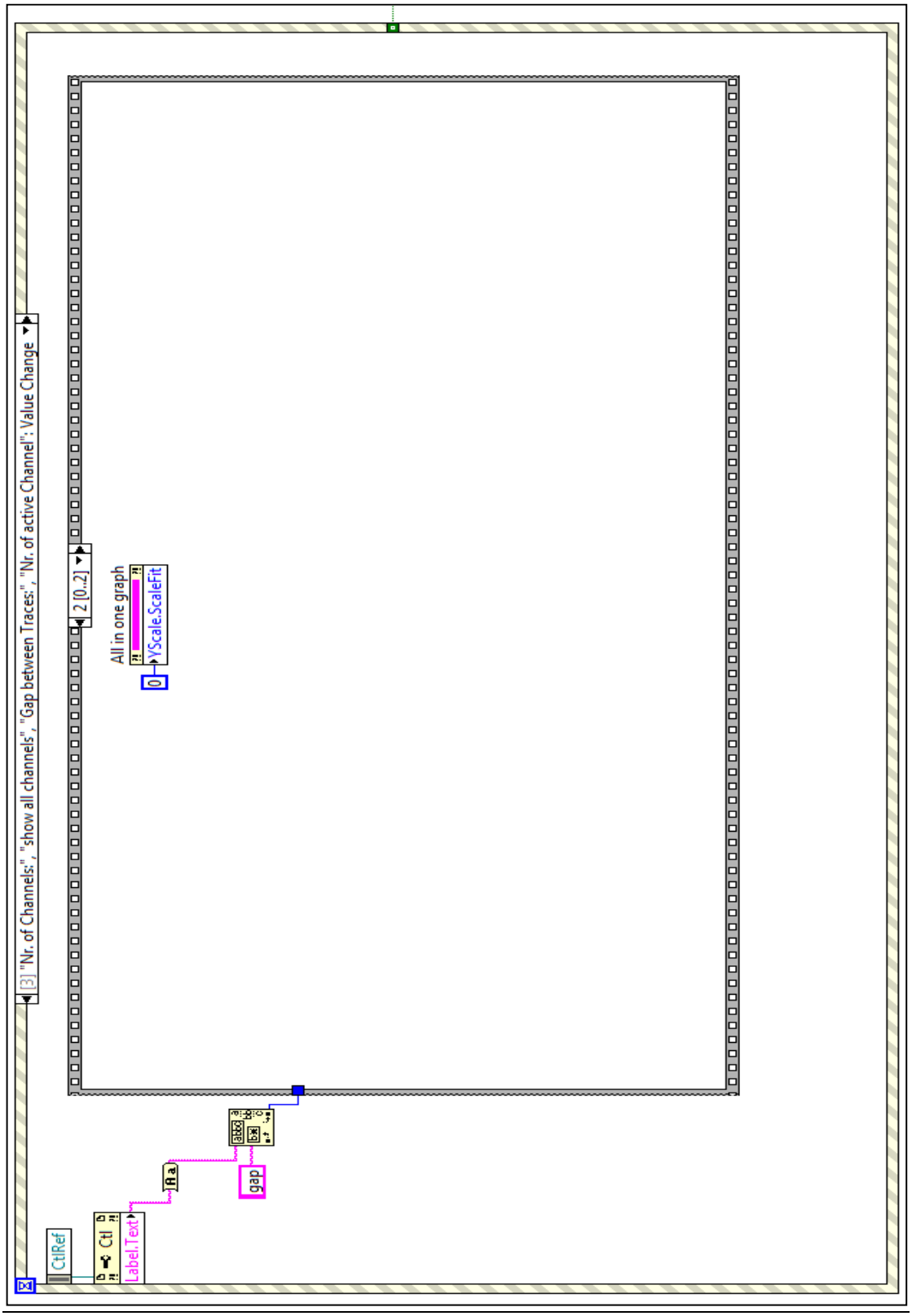
"Export event":



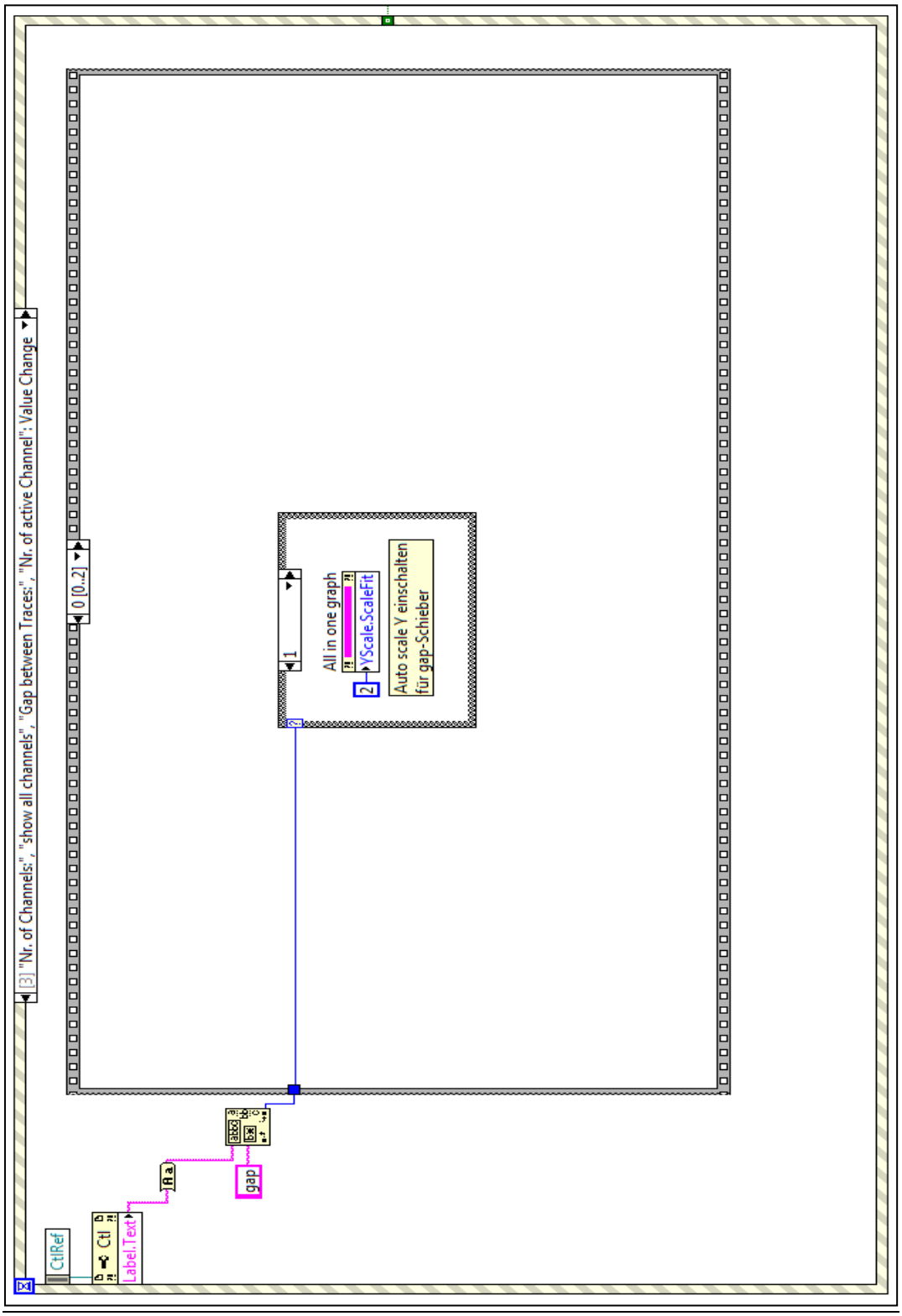
"All in one graph event":



"Auto-scaling event":



"Auto-scaling during gaps event":



# Appendix B: Python Code

**Python code for the noise and "Gaussian signal":**

```
# -*- coding: utf-8 -*-

from __future__ import division
"""
Created on Fri Aug 09 09:58:44 2013

@author: k.abdel-latif
"""
#from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt
from numpy import mean
from scipy import signal
import scipy
import math
from numpy import Inf
from time import gmtime, strftime
#import matplotlib.mlab as mlab
dt = 0.01
#Time till 500 ms
t = np.arange(0, 500, dt)
noise = np.random.randn(len(t))                 # random noise
#Calculating the Root-Mean-Square (RMS)
RmsValue =sqrt((1/len(noise))*mean(noise**2))
#Calculating the standard deviation
Stdev=np.std(noise)
upperstd=3*Stdev
lowerstd=-3*Stdev
upperlimit=RmsValue+upperstd
lowerlimit=RmsValue+lowerstd
#function for Max & Min peak finder
def max_fn(DataFlow,ThresholdValue):
    return max(v for v in DataFlow if v<ThresholdValue)
```

```
maxpeak=max_fn(noise,upperlimit)
print "maximum peak in range is : ", maxpeak
def min_fn(DataFlow,ThresholdValue):
    return min(v for v in DataFlow if v>ThresholdValue)
minpeak=min_fn(noise,lowerlimit)
print "minimum peak in range is : ", minpeak
MxpeakOfSignal=max_fn(gauss,Inf)
print "maximum peak of the signal : " , MxpeakOfSignal
FWHM = 2**math.sqrt(2*math.log(2))*sigma
print "FWHM of Signal = " ,FWHM
if FWHM < 10 and MxpeakOfSignal > maxpeak:
    print "There is a AP signal at ",strftime("%H:%M:%S", gmtime())
#Calculating Peak to peak Value:
Peak_to_Peak = maxpeak-minpeak
print "The value of Peak to Peak ( from electrode noise ) is : " ,
Peak_to_Peak
plt.plot(t, noise, 'b-')
plt.xlim(0,500)
plt.xlabel('Time in ms')
plt.ylabel('Amplitude in mV')
plt.grid(True)
rmsline = plt.axhline(y=RmsValue,linewidth=2, color='r')
upperlimitline=plt.axhline(y=upperlimit,linewidth=2, color='g')
lowerlimitline=plt.axhline(y=lowerlimit,linewidth=2, color='g')
MxpeakOfSignalLine=plt.axhline(y=MxpeakOfSignal,linewidth=2,
color='black',)
plt.text(500, RmsValue, "RMS Value", color='r')
plt.text(500,upperlimit , "3*Sigma", color='g')
plt.text(500,lowerlimit , "-3*Sigma", color='g')
plt.text(500,MxpeakOfSignal , "Max peak of signal", color='black')
plt.plot(gauss, color='y')
plt.show()
```

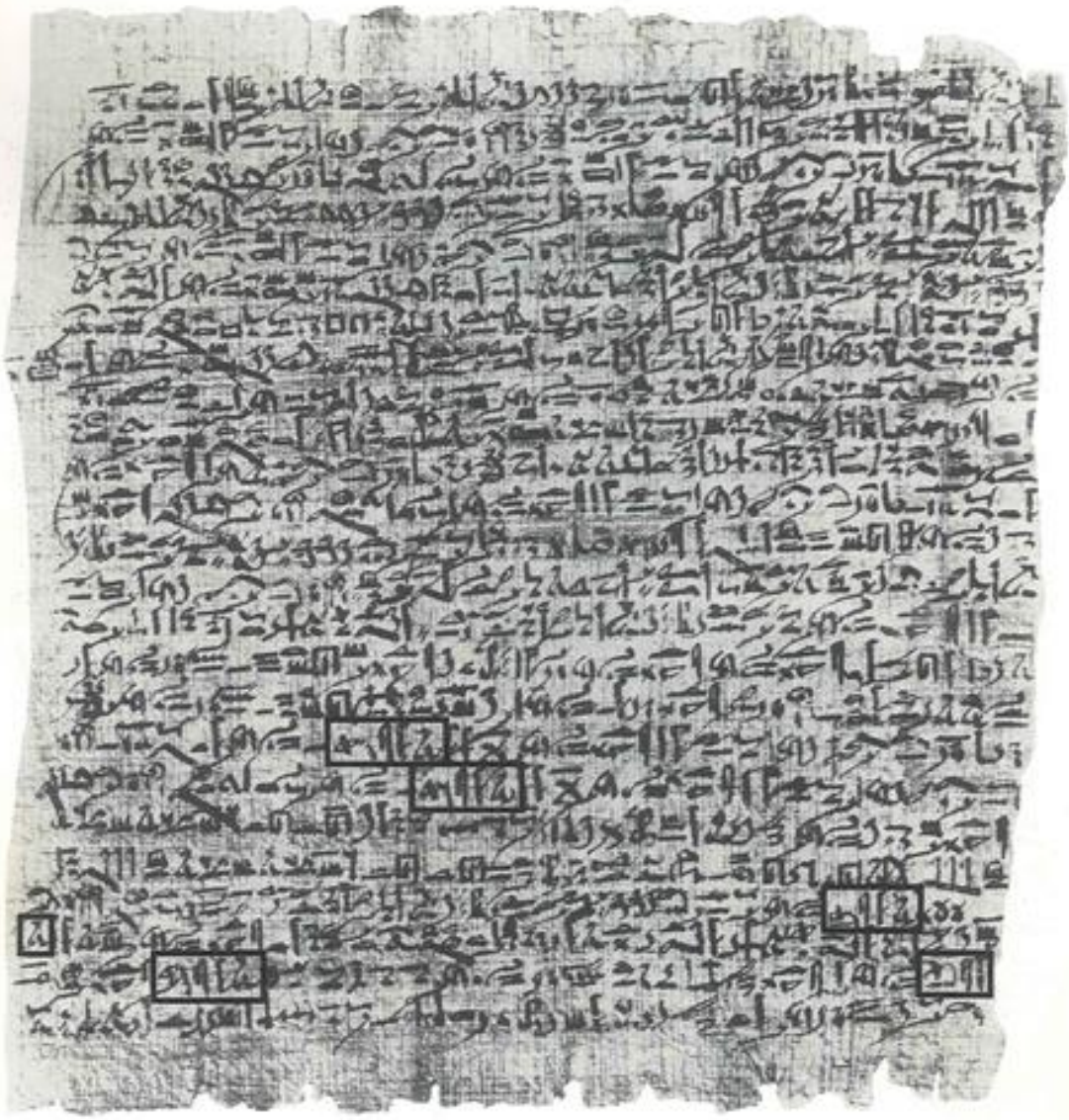
# Appendix C: Cell Culture for HL-1 cell-line

For the cultivation of the cardiomyocyte-like cell HL-1:

First, the MEA chips were cleaned for two hours in flowing ultrapure water. They were then sterilized for 30 minutes with UV (ultraviolet) light. The surface of the substrates was then coated with "Fibronectin" with a concentration of 1 ml in 200 µl of 0.02% Bacto TM Gelatin (Fisher Scientific) for an incubation time of 1 hour.

Confluent HL-1 cells in a T-25 flask were treated with 0.025% Trypsin/EDTA, suspended in 5 ml of "Claycomb" medium and centrifuged for 5 minutes at 1700 rpm. The pellet was then re-suspended in 3 ml of medium and 30 µl were plated on the substrates. 1 ml of medium was added after 15 minutes, and the substrates were incubated for 3 days until the cells formed a confluent monolayer on the lines and started to contract.

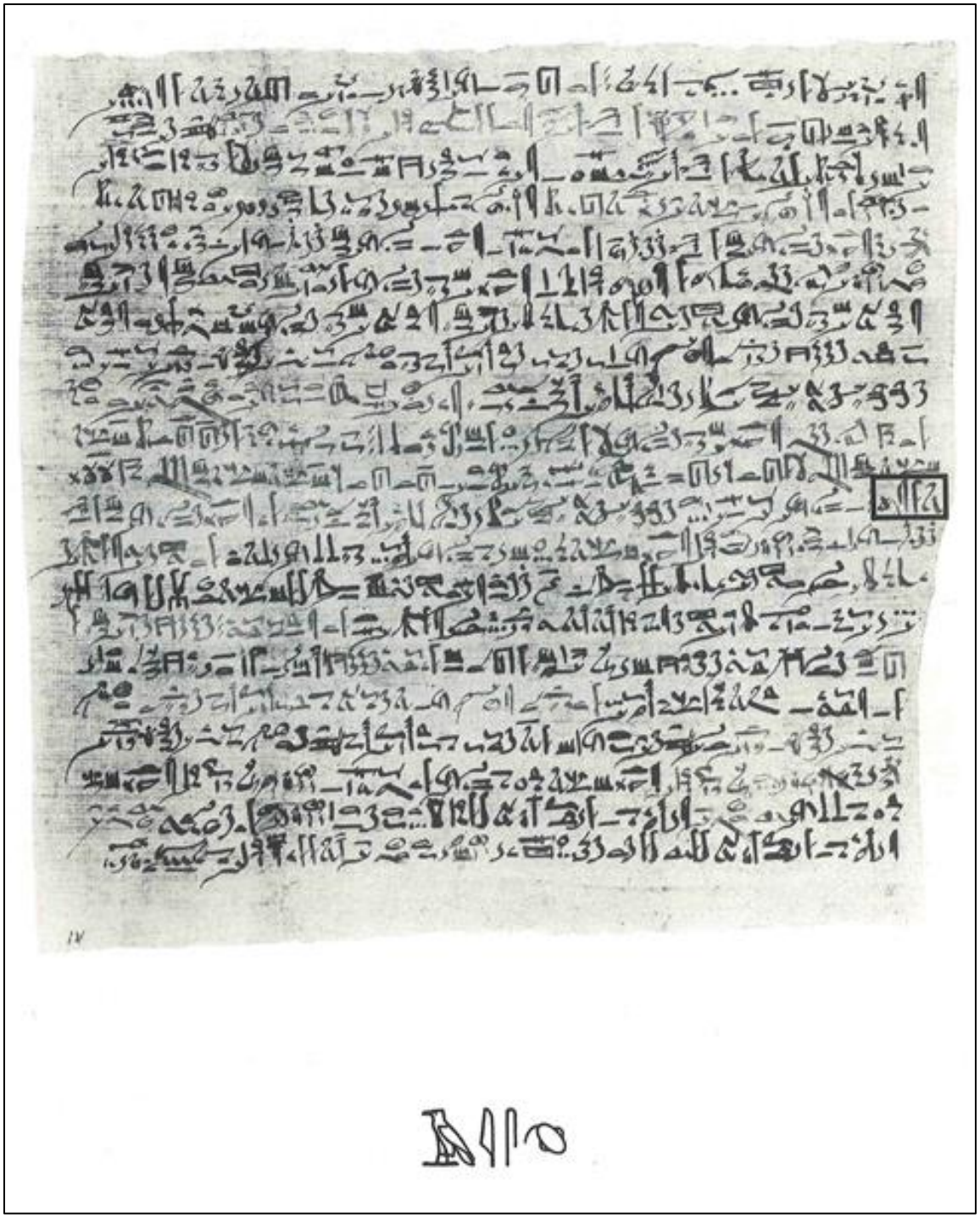
# Appendix D: The Edwin Smith Surgical Papyrus



Columns II (left) and IV (right) of the Edwin Smith Surgical Papyrus

This papryus, written in the Seventeenth Century B.C., contains the earliest reference to the brain anywhere in human records. According to James Breasted, who translated and published the document in 1930, the word brain 𓆎 (*ỉs*) occurs only 8 times in ancient Egyptian, 6 of them on these pages of the Smith Papyrus describing the symptoms, diagnosis and prognosis of two patients, wounded in the head, who had compound fractures of the skull. The entire treatise is now in the Rare Book Room of the New York Academy of Medicine.

*Reference:* Breasted, James Henry. The Edwin Smith Surgical Papyrus, 2 volumes. The University of Chicago Press, Chicago. 1930.



# Acknowledgements

I would like to express my sincere gratitude to my advisor **Prof. Dr. Michael J. Schöning** for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge.

I would like to thank **Prof. Dr. Bernhard Wolfrum** for his encouragement, insightful comments, and hard questions. His guidance helped me in all the time of research and writing of this work.

My sincere thanks also goes to **Prof. Dr. Andreas Offenhäusser** for offering me the opportunity to work on diverse exciting projects.

I would like to express my deepest appreciation to all those who provided me the possibility to complete this work.

A special gratitude I give to **Jan Schnitker** whose contribution in stimulating suggestions and encouragement helped me to coordinate my project. From the first day at work until the end of thesis, he was always there to guide and help me. I would also thank him for the excellent supervision. A special thanks for the guidance during writing this report.

Special thanks to **Dieter Lomparski** for his great help for teaching me many things in the LabVIEW programming language. I would like to thank him for the guidance during the development of the viewer program and being there whenever I needed help.

I would also like to thank the following persons for supporting me during my Bachelor thesis:

**Wienke Lange:** For the guidance in the labs, the great encouragement and support during the project and the writing of the thesis.

**Anna Czeschik:** For helping me with Python programming language.

**Francesca Santoro:** For showing me the cell culture procedure.

**Philipp Rinklin:** For his great support during the writing of the thesis.

**Cony Herrera:** For showing me the encapsulation of the MEA chips and the support during the writing of the thesis.

I thank my fellow lab mates at Forschungszentrum: **Sung-Eun, Susanne, Jonas, Alexey**, **Koji**, **Enno**, **Volker**, **Tianyu**, **Vanessa**, **Pinggui**, **Jing** for the stimulating discussions, and for all the fun we have had in the last five months.

Of course, I would also like to thank my family and friends for supporting me and being there whenever I needed help.

Jül-4366
September 2013
ISSN 0944-2952